The Geometric Design of Functional Shapes

by

Todd Robert Jackson

B.S.E., Aerospace Engineering (1994)
Princeton University

Submitted to the Department of Ocean Engineering
in partial fulfillment of the requirements for the degrees of
Master of Science in Naval Architecture and Marine Engineering
and
Master of Science in Mechanical Engineering
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 1997

Author ....
Department of Ocean Engineering
January 8, 1997

Certified by ....
Nicholas M. Patrikalakis
Professor of Ocean Engineering
Thesis Supervisor

Certified by ....
David C. Gossard
Professor of Mechanical Engineering
Thesis Reader

Accepted by ....
J. Kim Vandiver
Chairman, Committee on Graduate Students

Accepted by ....
Ain A. Sonin
Chairman, Committee on Graduate Students

The Geometric Design of Functional Shapes

by

Todd Robert Jackson

Submitted to the Department of Ocean Engineering
on January 8, 1997, in partial fulfillment of the
requirements for the degrees of
Master of Science in Naval Architecture and Marine Engineering
and
Master of Science in Mechanical Engineering

Abstract

The design of geometric shapes subject to physical constraints, such as the generation of lift from hydrofoils, remains an important problem in Computer Aided Geometric Design (CAGD). In such design problems, the quality of the shape is judged by how well it satisfies certain physical constants rather than just its geometric properties. In this thesis, two variations of the functional design of lifting shapes will be considered. First, a method to functionally design a mean camber surface (3D) from hydrodynamic data generated by a propeller analysis code will be presented. This method involves the approximation of a grid of position vectors and the kinematic boundary condition sampled over an initial blade shape. To linearize the problem, the physical constraint is formulated in terms of a constraint on the surface normal allowing for a direct solution with linear equation system solvers. Next, an algorithm to functionally design the mean camber line of a foil shape (2D) is presented in which an interpolation approach is used to satisfy the kinematic boundary condition at a set of points along a foil's mean camber line. Given a coefficient of lift, circulation distribution, and desired order of the curve, the algorithm "evolves" a flat line described in terms of a B-spline curve into a mean camber line carrying the prescribed loading. Examples of the two methods are given. Possible related applications for fitting curves and surfaces to sets of data points and normal vectors are also presented.

Thesis Supervisor: Nicholas M. Patrikalakis
Title: Professor of Ocean Engineering

Thesis Reader: David C. Gossard
Title: Professor of Mechanical Engineering

Acknowledgements

My work on this thesis has benefitted from the support and guidance of several people in the field of CAD whom I wish to acknowledge. First, I would like to thank my advisor, Professor Nicholas M. Patrikalakis, for his guidance, support, and the education he has given me in the area Computer Aided Design and spline representations. Another individual to whom I am deeply grateful is Dr. Xiuzi Ye who was instrumental in helping me get started with this research and was always able to point me in a direction to help me get past the inevitable obstacle as it came along.

I would also like to thank Professor Justin E. Kerwin and his students Dr. Todd Taylor and Mr. Scott Black for providing me access to and instruction on their propeller design codes as well as the many insightful discussions about potential flow theory and the design methods for lifting surfaces. Mr. Stephen L. Abrams, Dr. Seamus T. Tuohy, and Dr. Takashi Maekawa from the M.I.T. Design Laboratory were also very helpful in answering my questions pertaining directly to the theory behind this research as well as the process of its implementation on the computer. I am also indebted to them for their patience. Discussions about hydrofoil analysis with Mr. William Millewski provided me with new perspectives for looking at the design of lifting surfaces and forced me to think about design in new ways. I also would like to thank my thesis reader, Professor David C. Gossard for his comments. Finally, support for this work was obtained in part from the Office of Naval Research (grant numbers N00014-94-1-1001 and N00014-96-1-0857) and from the Naval Sea Systems Command (grant numbers N0002495WR10235 and N0002496WR10553), which is gratefully acknowledged.

Contents

List of Figures

List of Tables

Nomenclature

$B_i^k, \bar{B}_j^l$	= i^{th}, j^{th} B-spline basis function of order k, l
C^k	= k^{th} order of continuity
C_L	= coefficient of lift
C_l	= coefficient of lift per unit span
E	= error in approximation
$E_{\vec{Q}}$	= error in approximating position data
$E_{\hat{n}}$	= error in approximating normal constraints
$\mathbf{I}$	= identity matrix
L	= lift
$\mathbf{M}$	= matrix coefficient
$O(f(x))$	= asymptotically upper bounded rate of growth
$\vec{P}(u)$	= parametric curve
$\vec{P}_i$	= the i^{th} control point or control vertex of a B-spline curve
$\vec{S}(u,v)$	= bi-parametric surface
$\vec{S}_{ij}$	= the $(i,j)^{th}$ control point or control vertex of a B-spline surface
$\vec{Q}_i, \vec{Q}_{ij}$	= node point on the mean camber curve, surface in the current iteration
$\vec{Q}_i^*, \vec{Q}_{ij}^*$	= node point on the mean camber curve, surface in the previous iteration
$T_{\vec{S}(u_{i_p}, v_{j_p})}$	= tangent plane of surface $\vec{S}(u,v)$ at u_{i_p}, v_{j_p}
$\mathbf{U}, \mathbf{V}$	= knot vectors in $\hat{u}, \hat{v}$ directions
$\vec{V}$	= velocity of fluid
V_∞	= free stream velocity of flow in $\hat{\imath}$ direction
k, l	= order of B-spline geometry in $\hat{u}, \hat{v}$ directions
m, n	= number of control vertices in $\hat{u}, \hat{v}$ directions
m_p, n_p	= number of data points in $\hat{u}, \hat{v}$ directions
$\hat{n}$	= normal to curve, surface in the current iteration
$\hat{n}^*$	= normal to curve, surface in the previous iteration
p	= pressure in fluid
$\hat{t}$	= tangent of curve
u, v	= coordinates in parametric space
u_i^*, v_j^*	= parametric coordinates of sampled point of B-spline geometry
u_i, v_j	= knot values in knot vectors $\mathbf{U}, \mathbf{V}$
x, y, z	= coordinates in physical space
w	= weight determining influence of terms in least-squares fit
Γ	= vorticity strength
γ_i	= lumped vortex strength
$\gamma(u)$	= circulation or vorticity distribution
ρ	= density of fluid

Chapter 1

Introduction

In Computer Aided Geometric Design (CAGD) and Computer Aided Design (CAD), B-spline curves and surfaces (polynomial and rational) are the most widely used for free-form applications [14]. Mainly controlled by their control vertices, a B-spline curve is an approximation to its control polygon and a surface is an approximation to its control mesh. Both can be used for interpolating or approximating (e.g. least-square fitting) grid points [14]. In addition, surfaces can be generated from curves by using skinning, cross-sectional design, or feature lines techniques [14, 4, 41].

Currently, B-splines are used mainly for *shape representation*, but not as widely and directly for *shape design* [7]. In the context of *shape representation*, only geometric data and no underlying physical constraints are considered. The use of B-spline curves and surfaces in this context is a pure mathematical representation which has many advantages such as geometrical intuitiveness, rich representation capability, and data reduction. In the context of *shape design*, however, geometric shapes are intended to serve certain functions. They are designed according to their underlying physical constraints such as hydrodynamic and aerodynamic constraints. For example, from the hydrodynamics point of view, the essential problem in designing a propeller blade is to find a so-called mean camber surface which carries a prescribed load distribution such that the kinematic hydrodynamic boundary condition [22]

$$\vec{V} \cdot \hat{n} = 0 \tag{1.1}$$

is satisfied on that surface in the presence of a given flow, where $\vec{V}$ is the velocity of the flow and $\hat{n}$ is the surface normal vector. In this thesis, we use the term *functional design* instead of *shape design.*

Functional design is different from fair curve or surface design. The quality of the former design is judged by the functional performance, whereas the latter one by fairness criteria [8, 28, 37]. The former one involves physical measures such as energy efficiency, whereas the latter one involves geometric measures such as curvatures. Nevertheless, they are sometimes related since fairness criteria frequently represent physical measures (such as thin plate energy [28]). However, it is possible to have curves or surfaces with fair geometries but which perform poorly.

Functional surface design is widely used in engineering. In fact, every engineering design is a functional design which serves certain application purposes. Examples include the blade surface design method developed by Kerwin *et al.* [22] and the design of ship hull forms developed by Nowacki [26]. In both methods, the design of the shape of the surfaces is coupled with flow analysis.

Although sometimes involving curvatures or higher order derivatives of surfaces, physical constraints for functional surface design frequently involve only surface normal vectors. The hydrodynamic constraints for propeller blade surfaces and the aerodynamic constraints for airfoils are two examples for such kind of constraints.

This thesis focuses on functional design of surfaces and curves due to physical constraints. Specifically, the design of mean camber lines and surfaces of hydrofoil and propeller blade shapes is studied. The performance of the shape influences the design directly through the incorporation of the physical constraints into the design process. The hydrodynamic constraint considered here is the kinematic boundary condition which involves the velocity of the fluid and the geometry (position and tangent plane) of the shape. First, a new method to functionally fit the mean camber surfaces of propeller blades to a set of position data and velocity vectors is presented. In this method, the non-linear physical constraints imposed on the surface are converted to constraints on the surface normal vectors at the given node points. The modified surface is generated by a weighted least-squares fitting of the grid points and the normal vectors at these points. This fitting problem is then converted to a linear problem with a larger number of equations by means of vector calculus, significantly reducing the complexity of the problem and saving computation time. As an application example, the functional design of propeller blade surfaces is studied. Next, a functional design method to design the mean camber line of a hydrofoil is considered. In this method, an interpolation approach is used to gradually evolve the shape of a B-spline curve from a straight line into a mean camber line producing the specified coefficient of lift and circulation distribution. For a given loading on the foil, the flow is evaluated at a set of points and a new curve is constructed, parallel to the sampled velocity vectors. The degrees of freedom in this design process are reduced to permit interpolation and a scaling step is used to help achieve convergence.

The remainder of this thesis is organized as follows.

First, in Chapter 1, relevant background material will be covered, including an overview of the physics behind the generation of lift and B-spline curve and surface representation. This will be followed by a brief literature review of past work on the design of functional shapes in Chapter 2.

In Chaper 3, the functional design of mean camber surfaces of propeller blades will be introduced. The kinematic boundary condition is first formulated as a non-linear physical constraint which is then converted into a constraint on the surface normal vectors at given node points. Furthermore, it is shown how this type of non-linear problem can be converted to a linear problem. A method to prescribe the surface normal vectors from other given physical field vectors, such as flow velocity vectors at the node points, is also developed. A least-squares fitting method of B-spline surfaces with normal vector constraints is then presented and extended to weighted least-squares fitting in an effort to better approximate the boundary curves such as the leading edges of propeller blades. Next, some examples of functional surface design by using the new method together with a propeller blade analysis program are presented. Differences between surface fitting of only position vectors and fitting of position and normal vectors with different weights are discussed, illustrating the influence of the normal vector constraints on the performance measures.

After the explanation of the least-squares methods of designing functional surfaces, an interpolation method to functionally design the mean camber lines of hydrofoils is developed in Chapter 4. This approach uses the physical constraint imposed by the kinematic boundary condition, but also requires that the grid of node points be repositioned such that they lie along lines normal to the previous curve and pass through their previous positions, thereby reducing the number of unknowns. In this manner, the number of equations can be balanced with the number of constraints and a solution can be found. The design process begins with a line described in terms of a B-spline curve of arbitrary order. The coefficient of lift and circulation distribution is also specified. Starting from zero, the loading on the foil is gradually increased, dictating modifications in the foil's shape to satisfy the boundary condition until the desired loading is reached. The result is a mean camber curve for a hydrofoil producing the specified coefficient of lift and lift distribution. Examples from the implementation of this design method illustrate the process.

Finally, in Chapter 5, some possible related applications for the fitting methods discussed in this thesis are presented. These include the shape preserving interpolation of sets of data and the approximation of constant distance offset curves.

In Chapter 6, conclusions and recommendations concerning this research are presented. The contributions from this thesis to the CAD community are summarized and discussed, followed by suggestions for future work in this area.

Chapter 2

Background

2.1 Introduction and background

This thesis deals with the design of mean camber lines and surfaces of lifting foils described in terms of B-splines. Before the actual research can be presented, some background material must be reviewed. First, the physics behind how lift is generated will be discussed. Potential flow will then be introduced as a means to represent ideal fluid flow passed a foil. Next, the geometric representation used to represent and manipulate the shape of the foil will be introduced. Finally, the inverse design process will be described, through which one designs the shape of a foil by specifying the desired lift distribution over the foil's geometry.

2.2 The physics behind the generation of lift

As a body moves through a fluid, it experiences forces generated by the pressure of the fluid as it flows around the body. The force exerted on a body can be resolved into two components, lift and drag. Lift is defined as the force acting normal to the body's direction of travel while drag is the force acting parallel and opposite to its motion. As the object moves through a fluid, the fluid's motion (flow) is dictated by the laws of physics and the geometry of the body. By creating an asymmetric flow as it travels, unbalanced regions of high pressure and low pressure will occur over the body creating a pressure difference across it. This pressure difference acts over the surfaces of the body to generate a net force. The generation of force due to a pressure imbalance is illustrated in Figure 2-1a.

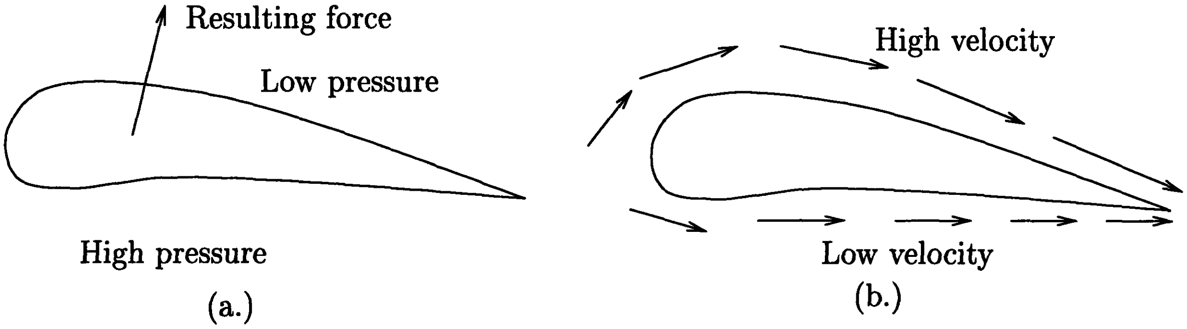


Figure 2-1: Generation of hydrodynamic force due to pressure difference.

In the early eighteenth century, Bernoulli postulated an equation relating the total pressure exerted by an inviscid, incompressible, steady fluid flow on its surroundings to the velocity at which it is flowing [25, pp. 107-8]. This relation has become known as Bernoulli's equation and relates the pressure in an inviscid, incompressible, steady flow to the local velocity as given by

$$p + \frac{1}{2}\rho V^2 = constant, \tag{2.1}$$

where V is the magnitude of the local flow velocity. Therefore, to generate a given pressure distribution, one can expect a corresponding velocity field as defined by Equation 2.1. As a fluid's velocity increases, the pressure it exerts on its surroundings decreases and vice verca. A simple illustration of the velocity field which might have generated the pressure distribution in Figure 2-1a is shown in Figure 2-1b. The fluid particles traveling over the top of the foil need to travel faster to keep up with their neighboring particles traveling along the bottom side of the foil. As a result, the flow over the top of the foil travels relatively faster than the flow over the bottom resulting in a pressure difference across the foil. As previously described, this pressure difference acting on the foil's surface will generate a force acting on the foil resulting in lift and drag.

In addition to the assumptions that the flow is inviscid and incompressible, if we also assume that the flow is irrotational, the flow can be represented by potential flow theory. With potential flow theory, the flow is modeled by distributing singularities outside of the fluid domain. The flow needs to satisfy kinematic and dynamic conditions along the boundaries of the fluid and the requirement that the velocity of the fluid remains finite. The singularities consist of sources, sinks, dipoles, and vortices. The work presented in this thesis involves vortex singularities, the kinematic boundary condition, and the requirement that the velocity of the fluid remains finite, as given by the Kutta condition.

A single vortex filament can be visualized as a whirlwind in the fluid. In two dimensions, the fluid flow reduces to concentric circular paths about the vortex as given by

$$\vec{V} = \frac{\Gamma}{2\pi r}\hat{\theta} \tag{2.2}$$

where Γ is the vortex strength, r is the distance from the center of the vortex to a point in the fluid, and $\hat{\theta}$ is the unit vector in the tangential direction about the point [38, p. 198].

This can be extended to three dimensions where the flow circulates about a vortex filament or space curve C [25, p. 191]. The velocity at any point in the fluid is evaluated by integrating along the entire length of the vortex, as given by

$$\vec{V} = -\frac{\Gamma}{4\pi}\int_C \frac{\vec{R} \times d\vec{l}}{R^3} \tag{2.3}$$

where $\vec{R}$ is a vector pointing from the differential element $d\vec{l}$ of the curve C to a point in space where we wish to calculate the velocity and $R = |\vec{R}|$. The ability to superpose flows in potential flow theory allows the flow about a foil or blade to be represented as the superposition a uniform flow with a distribution vorticity over the mean camber line/surface.

The kinematic boundary condition, the first physical constraint in this design problem, states that the velocity of an ideal, inviscid fluid along a solid boundary must be parallel to the boundary. Mathematically, this can be expressed as a requirement on the scalar product of the fluid velocity $\vec{V}$ with the normal of the boundary of the fluid domain $\hat{n}$ as

$$\vec{V} \cdot \hat{n} = 0 \quad (\textit{over a stationary body}). \tag{2.4}$$

Intuitively, this makes sense since we know from experience that fluid cannot flow through a solid boundary, hence the component of the fluid velocity parallel to a boundary's normal must vanish.

The Kutta condition is the second physical constraint which must be considered in the functional design of foils. In any real flow, the velocity of the fluid at any point must remain finite. This creates a special condition at any sharp edge along the boundary of the fluid, known as the Kutta condition. Since a sharp edge presents no radius for the fluid to flow around, infinite velocities will occur if the flow does not leave the edge smoothly (i.e. parallel to the boundaries). Therefore in order for the velocity to remain finite, the Kutta condition requires that the flow leave any sharp edge smoothly.

By strategically positioning the vortices along the mean camber line/surface of a foil in a uniform flow and adjusting their strengths such that the kinematic boundary condition and the Kutta condition are satisfied, the flow about the foil can be modeled. A conceptual illustration of this is shown in Figure 2-2. The flow about the foil is equivalent to the flow about a distribution of vorticity/circulation along the mean camber line placed in a uniform stream.

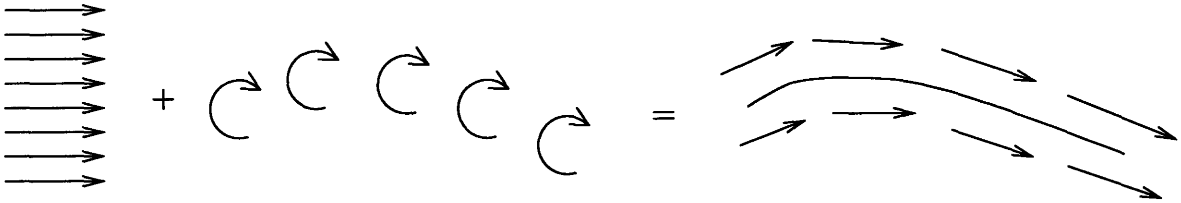


Figure 2-2: Representation of ideal flow over mean camber line with potential flow theory.

In three dimensional flow, the vorticity becomes a vector quantity $\vec{\omega}$ and the total vorticity in the fluid must be conserved, as required by the Helmholtz equation shown in Equation 2.5 [42, p. 90].

$$\frac{D\vec{\omega}}{Dt} = (\vec{\omega} \cdot \vec{\nabla})\vec{V} + \nu \nabla^2 \omega \tag{2.5}$$

where $\frac{D}{Dt}$ denotes the substantial derivative, ∇ denotes the del or gradient operator, ∇^2 denotes the Laplacian, and ν is the kinematic viscosity. In representing a 3D lifting surface, both the kimematic boundary condition and the conservation of vorticity must be satisfied over the surface. In addition since the Helmholtz equation states that a vortex filament may not begin or end in a fluid, vorticity leaving the lifting the surface must leave as a trailing wake, carrying the vorticity off to infinity.

2.3 Overview of B-spline representation

Before discussing the functional design of shapes, a method of storing, manipulating, and interrogating the shape of a lifting surface must be introduced. Ideally, the method of shape representation would efficiently provide all of these aforementioned functions with minimal memory and computational time requirements. For purposes of shape representation, B-spline representation has been chosen, providing a compact representation of the shape of the curves and surfaces to be functionally designed, while still providing efficient methods to evaluate, interrogate, and manipulate these shapes. Due to these desirable properties, B-spline geometry representation has grown in popularity to the point where it has become an industry standard and is included on most CAD, CAM, and FEM packages [11, 14, 15]. By choosing to follow the industry standard, integration of the design method presented here with current CAD packages is possible, making this research more applicably to industry.

The B-spline representation of a shape, the curve or surface is described in terms of control vertices and blending (basis) functions. In the simplest and most intuitive sense, a B-spline curve or surface can be considered an approximation to a polygon or net of control vertices, respectively. For a more complete understanding of B-splines, the B-spline basis functions and how they are used to evaluate a B-spline curve or surface must first be explored. For a full description of B-spline representation, the reader should consult [2, 11, 14, 20, 35].

2.3.1 B-spline basis functions

The B-spline basis functions are used as blending functions which determine the influence which each control vertex has on a point of a curve or surface at a given parametric value. Each basis function of a given order k is a piecewise polynomial of degree $k-1$ and is defined recursively with respect to a knot vector $\mathbf{U}$ according to Equations 2.6-2.7 [20].

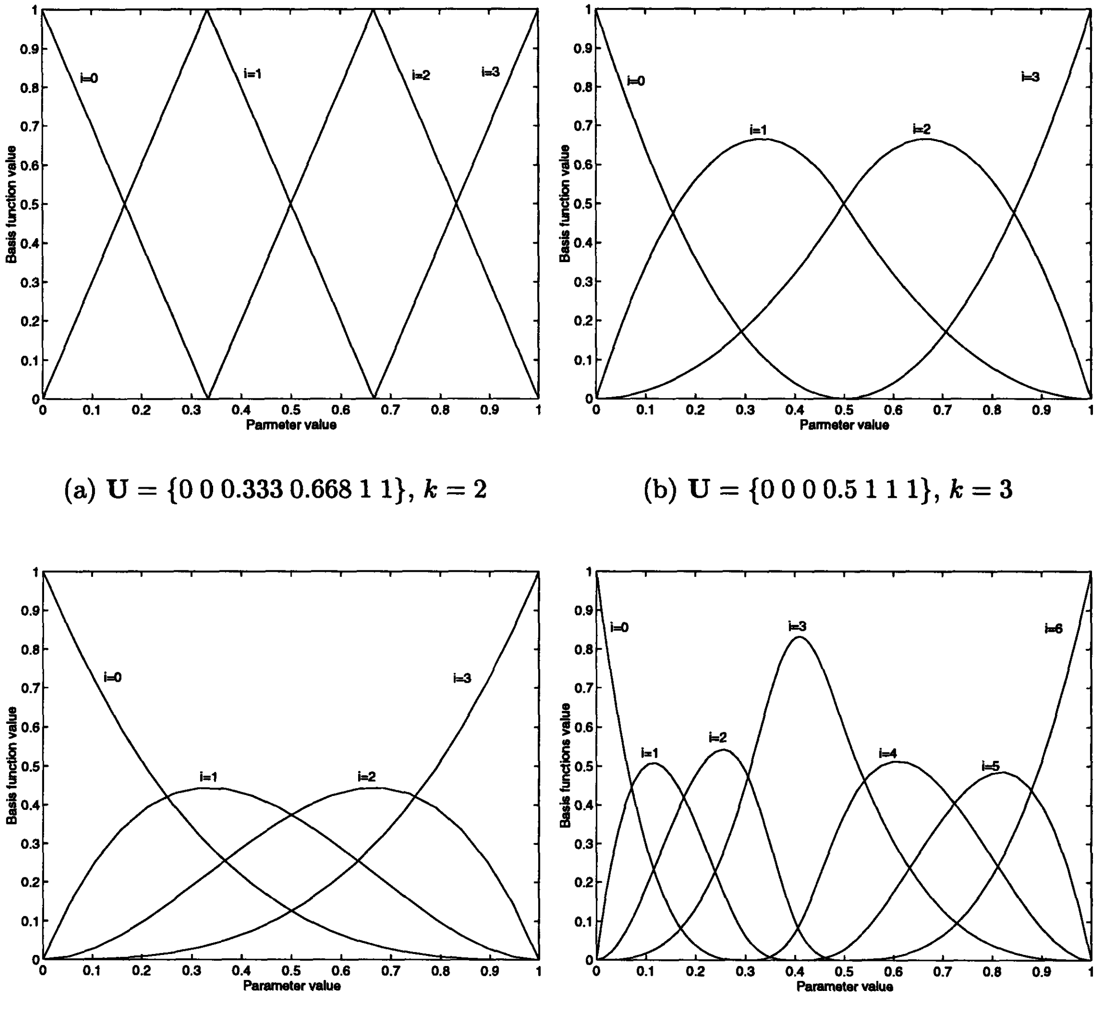
$$B_i^1(u) = \begin{cases} 1 & \text{if } u_i \le u < u_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{2.6}$$

$$B_i^k(u) = \frac{u-u_i}{u_{i+k-1}-u_i} B_i^{k-1}(u) + \frac{u_{i+k}-u}{u_{i+k}-u_{i+1}} B_{i+1}^{k-1}(u) \qquad \text{if } k > 1. \tag{2.7}$$

The knots in the knot vector $\mathbf{U}$ represent values at which discontinuities may exist. Each B-spline basis function is at least $k-p-1$ times continuously differentiable at the knots, where p is the knot multiplicity. By convention, the undefined ratio $\frac{0}{0}$ is taken to be zero thereby allowing the introduction of higher levels of discontinuities at a parametric value by repeating internal knot values. Several B-spline basis functions of varying orders are illustrated in Figures 2-3a–d. Note that the sum of all of the basis functions defined by a knot vector is unity and each individual basis function is non-negative for all values of u. From this observation, the B-spline basis functions can be considered to form a partition of unity.

2.3.2 B-spline curves

When the B-spline basis functions are used as blending functions with a polygon of control vertices, a B-spline curve can be formed. The influence of an individual control vertex $\vec{P_i}$ at a particular parametric value u along the curve is determined by the value of the control vertex's corresponding basis function $B_i^k(u)$. When the



(a) $\mathbf{U} = \{0\ 0\ 0.333\ 0.668\ 1\ 1\}$, $k = 2$

(b) $\mathbf{U} = \{0\ 0\ 0\ 0.5\ 1\ 1\ 1\}$, $k = 3$

(c) $\mathbf{U} = \{0\ 0\ 0\ 0\ 1\ 1\ 1\ 1\}$, $k = 4$

(d) $\mathbf{U} = \{0\ 0\ 0\ 0\ 0.3\ 0.4\ 0.5\ 1\ 1\ 1\ 1\}$, $k = 4$

Figure 2-3: Examples of several B-spline basis functions with different orders and knot vectors.

influences of all of the control vertices are combined, a point on the curve can be evaluated. Mathematically, this is written as

$$\vec{P}(u) = \sum_{i=0}^{m-1} \vec{P}_i B_i^k(u), \tag{2.8}$$

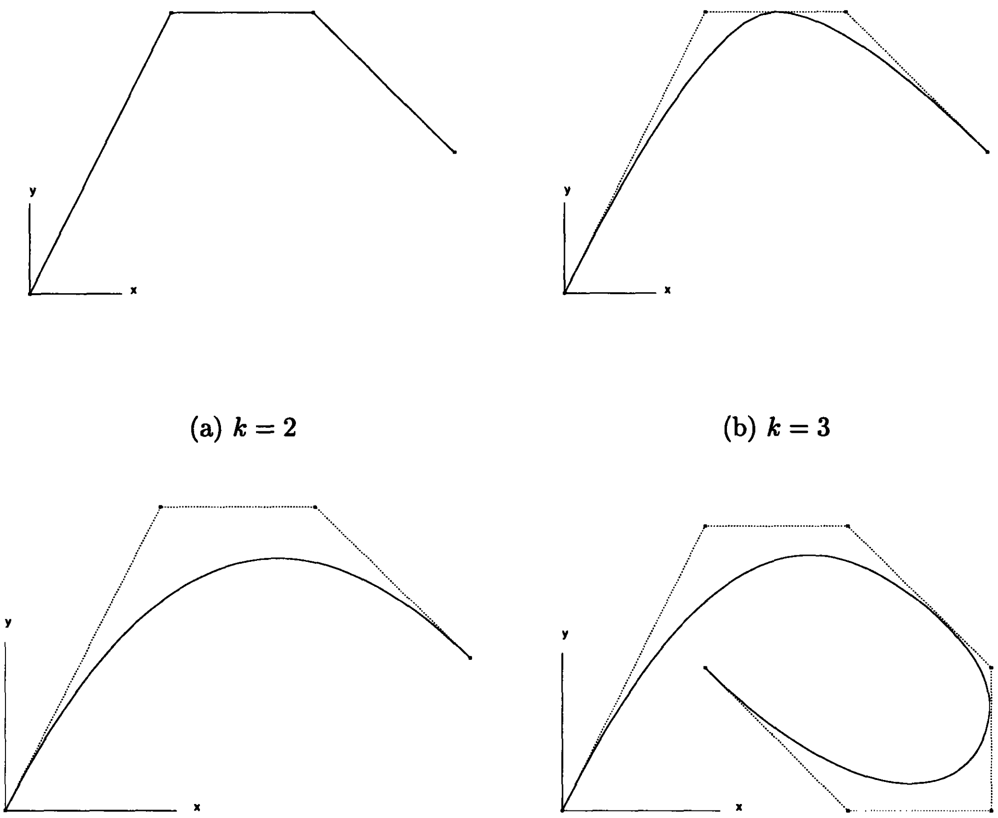
where m is the number of control vertices. Therefore, a B-spline curve is defined by m control vertices (defining a control polygon) and the corresponding m B-spline basis functions. The basis functions are of order k and are evaluated with respect to a knot vector. For the open B-spline curves considered in this thesis, each knot vector contains $m + k$ knots. Inheriting the continuity properties of the basis functions, a B-spline curve overall is at least C^{k-p-1} continuous, where p is the knot multiplicity. In between knot values, however, the curve is C^∞ continuous. Examples of B-spline curves defined in terms of the basis functions plotted in Figure 2-3a–d are shown in figures Figure 2-4a–d, respectively.

2.3.3 B-spline surfaces

A B-spline surface is a logical extension of a B-spline curve to a higher dimension. One can visualize a B-spline surface as the surface swept out by dragging a B-spline curve through space such that the paths followed by each of the control vertices are themselves B-spline curves. The surface $\vec{S}(u, v)$ is formally defined in Equation 2.9 by using a net of $m \times n$ control vertices $\vec{S}_{ij}$ and the basis functions $B_i^k(u)$ and $\bar{B}_j^l(v)$.

$$\vec{S}(u, v) = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \vec{S}_{ij} B_i^k(u) \bar{B}_j^l(v) \tag{2.9}$$

The basis functions are evaluated by the same recursive definition used for B-spline curves given in Equations 2.6-2.7 with each parametric direction assigned its own knot vector ($\{u_0\ u_1\ \ldots\ u_{m+k-1}\}$ and $\{v_0\ v_1\ \ldots\ v_{n+l-1}\}$). Similar to the curve's continuity properties, a B-spline surface will be at least C^{k-p_1-1} and C^{l-p_2-1} continuous in the u and v directions, respectively, where p_1 and p_2 are the knot multiplicities in the u and v knot vectors. An example of a B-spline surface and its control net are shown



(a) $k = 2$

(b) $k = 3$

(c) $k = 4$

(d) $k = 4$

Figure 2-4: Examples of several B-spline curves of different orders and knot vectors.

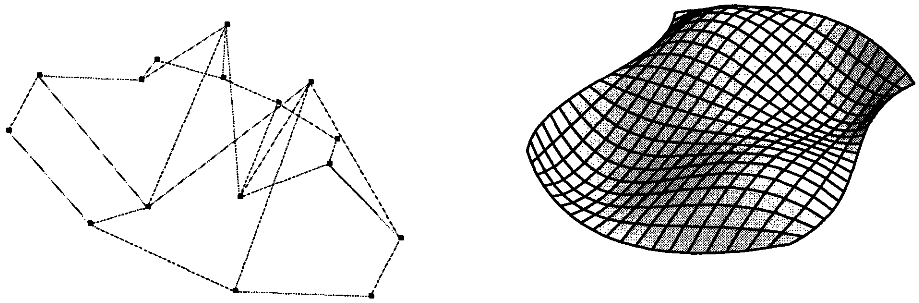


Figure 2-5: Example of a B-spline surface control net and the resulting surface.

in Figure 2-5. In the control net is translated for clarity.

2.3.4 Derivatives of B-splines

In order to facilitate functional design, the chosen shape representation must be able to provide ways of interrogate properties of the shape, such as evaluation of points at particular parameter values, curve/surface tangent(s), surface normals, and curvature. By using B-spline representation, derivatives of curves and surfaces can readily by evaluated by taking the derivative of the basis functions as shown in Equation 2.10 representing the derivative of a B-spline curve.

$$\vec{P}(u) = \sum_{i=0}^{m-1} \vec{P}_i B_i^k(u) \tag{2.10}$$

The first order derivative of the B-spline basis functions $B_i^k(u)$ is a combination of two lower order B-spline basis functions $B_i^{k-1}(u)$ and $B_{i+1}^{k-1}(u)$ as follows [2, 20, 35]:

$$\frac{dB_i^k(u)}{du} = (k-1)\left(\frac{B_i^{k-1}(u)}{u_{i+k-1} - u_i} + \frac{B_{i+1}^{k-1}(u)}{u_{i+k} - u_{i+1}}\right) \tag{2.11}$$

By repeatedly applying Equation 2.11, any order derivative can be evaluated allowing the direct evaluation of tangent planes, curvature, and normals at any point on a curve or surface.

2.3.5 Efficient implementation of B-spline representation

As was previously stated, it is desirable to be able to represent the shapes being designed with minimal computational and memory expense. The B-spline representation of geometries has been found through experience to provide sufficient shape representation flexibility with a relatively low number of control vertices for a wide range of applications [14]. Since the memory requirements for storage of the representation are directly proportional to the number of control vertices used in the representation ($O(m)$ for curves or $O(m \times n)$ for surfaces), a complete representation of a shape can be maintained with minimum overhead.

The process of evaluating and interrogating a B-spline geometry can also be performed with relatively little memory and computational requirements. First note that due to the local support of the basis functions, at most only k functions will be non-zero at any given parameter value u for functions of order k. Therefore, only k basis functions need to be evaluated. Furthermore, the recursive definition of the basis functions and their derivatives can readily be implemented into a *memoized* table. Memoization involves storing the answers to common subproblems of a larger problem so the answers can simply be looked up when they are encountered again [10]. By using memoization when evaluating the B-spline basis functions, the evaluation of each lower order/derivative basis function occurs only once, at which time it is stored in a table. For any subsequent computation involving that function, the value is simply looked up from the table rather than recomputed. The end result is that all of the basis functions can be evaluated in $O(k^2)$ time. Furthermore, any derivative of the basis functions can also be efficiently evaluated in the same manner. Since the basis functions are differentiable at most $k-1$ times, an algorithm to evaluate all derivatives of the B-spline basis functions can be executed in at most $O(k^3)$ time, requiring $O(k^3)$ memory. The end result is that any point on a B-spline shape can be interrogated in time asymptotically bounded by the cube of the order of the B-spline functions being used. The number of control vertices, representing degrees of freedom to represent shape, affect the storage requirements and not the computational time re-

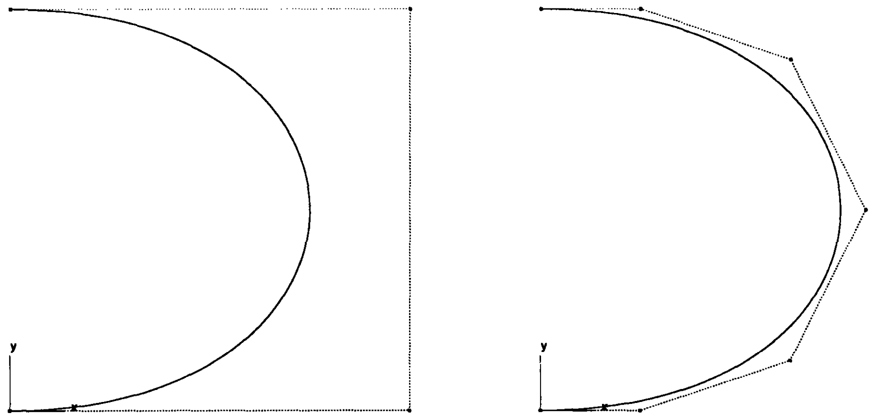


Figure 2-6: Insertion of a knot into a B-spline curve increases the degrees of freedom of the geometry without changing the shape.

quired to interrogate the shape at some particular value(s). The memoized algorithm used for the evaluation of the B-spline basis functions is given as Algorithm 1 [34].

2.3.6 Knot insertion: addition of degrees of freedom

Besides providing an efficient and compact form for representing a shape, the use of B-spline geometries also allows the addition of degrees of freedom to a geometry without inadvertently altering its shape. This can be accomplished by inserting a knot into the knot vector which introduces another control vertex to a curve's control polygon (or a row/column of control vertices in the case of a surface) [2, 20]. With the insertion of each new knot, the positions of the control vertices, along with any new control vertex, are recomputed so as to represent the original geometry exactly, as in Figure 2-6. In fact, since only at most k basis functions are affected by the value of any knot, the position of only the control vertices whose corresponding basis functions were affected by the new knot must be recomputed. With the ability to efficiently introduce new degrees of freedom, B-spline representation becomes an ideal choice for functional design since a shape with few degrees of freedom can be used initially and knots inserted where additional freedom is needed to satisfy the physical constraint as the shape converges to a solution. Knot insertion is documented as

Algorithm 1 Calculate $\frac{\partial^j B_i^k}{\partial u^j}$ for $0 \leq i < m$.

Require: k, order of basis functions; d, order of derivative; m, number of basis functions to be calculated; $\mathbf{U}$, knot vector of length $k + m$ containing non-decreasing elements; u, value at which functions are to be evaluated.

Ensure: $\mathbf{B}$ is an array of length m containing the desired derivatives of the B-spline basis functions of order k evaluated at parameter value u with respect to knot vector $\mathbf{U}$.

```
{Find the index of the non-zero, first order basis function.}
$i \leftarrow 0$
while $u \geq U_i$ do
    $i \leftarrow i + 1$
$i \leftarrow i - 1$
{Initialize the first column of the memoized table.}
$\mathbf{T}_{0,0} \leftarrow 1$ {Value of the $i^{th}$ (only non-zero) first order basis function.}
for $r = 1$ to $d$ do
    $\mathbf{T}_{r,0} \leftarrow 0$ {All derivatives of $1^{st}$ order function are zero.}
{Fill in first row of memoized table.}
for $c = 1$ to $k - 1$ do
    $i \leftarrow i - 1$
    for $r = 0$ to $c - 1$ do
        $\mathbf{T}_{0,(\frac{c(c+1)}{2}+r)} = \frac{u-\mathbf{U}_{i+r}}{\mathbf{U}_{i+r+c}-\mathbf{U}_{i+r}}\mathbf{T}_{0,(\frac{c(c-1)}{2}+r-1)} + \frac{\mathbf{U}_{i+r+c+1}-u}{\mathbf{U}_{i+r+c+1}-\mathbf{U}_{i+r+1}}\mathbf{T}_{0,\frac{c(c-1)}{2}+r}$
{Fill in higher order derivatives. (remaining rows)}
for $j = 1$ to $d$ do
    $i \leftarrow i + k - 1$
    for $c = 1$ to $k - 1$ do
        $i \leftarrow i - 1$
        for $r = 0$ to $c - 1$ do
            $\mathbf{T}_{0,\frac{c(c+1)}{2}+r} =$
                $\frac{(u-\mathbf{U}_{i+r})\mathbf{T}_{0,\frac{c(c-1)}{2}+r-1}+\mathbf{T}_{j-1,\frac{c(c-1)}{2}+r-1}}{\mathbf{U}_{i+r+c}-\mathbf{U}_{i+r}} +$
                $\frac{(\mathbf{U}_{i+r+c}-u)\mathbf{T}_{j,\frac{c(c-1)}{2}+r}-\mathbf{T}_{j-1,\frac{c(c-1)}{2}+r}}{\mathbf{U}_{i+r+c+1}-\mathbf{U}_{i+r+1}}$
{Place basis functions into array to be returned}
for $r = 0$ to $m - 1$ do
    if $r < i$ or $r > i + k - 1$ then
        $\mathbf{B}_r \leftarrow 0$
    else
        $\mathbf{B}_r \leftarrow \mathbf{T}_{d,\frac{k(k-1)}{2}+r-i}$
```

Algorithm 2 below [2].

Algorithm 2 Knot insertion into B-spline curve.

```
Require: $k$, order of curve; $m$, number of control vertices; $\mathbf{U}$, knot vector of length $k+m$ containing non-decreasing elements; $u^*$, knot to be inserted; $\vec{\mathbf{P}}$, array of control vertices.
Ensure: $\vec{P}^*(u) = \vec{P}(u)$, B-spline curve $\vec{P}^*(u)$ is defined by $m^*$ control vertices $\vec{\mathbf{P}}^*$ and knot vector $\mathbf{U}^*$.
$loc \leftarrow 0$
while $u^* \geq \mathbf{U}_{loc}$ do {Find location of knot in knot vector.}
    $loc \leftarrow loc + 1$
    $\mathbf{U}^*{}_{loc} \leftarrow \mathbf{U}_{loc}$ {Copy beginning knots}
$\mathbf{U}^*_{loc} = u^*$ {Insert new knot.}
for $i = loc$ to $m + k$ do
    $\mathbf{U}^*{}_{i+1} \leftarrow \mathbf{U}_i$ {Copy remaining knots.}
for $i = 0$ to $m + 1$ do {Compute new vertices.}
    if $i \leq loc - k$ then {Vertex unchanged.}
        $\vec{\mathbf{P}}^*_i \leftarrow \vec{\mathbf{P}}_i$
    else if $i \geq loc - k + 1$ and $i \leq loc$ then {Compute new vertex.}
        $\alpha \leftarrow \frac{u^* - u_i}{u_{i+k-1} - u_i}$
        $\vec{\mathbf{P}}^*_i \leftarrow \alpha\vec{\mathbf{P}}_i + (1-\alpha)\vec{\mathbf{P}}_{i-1}$
    else {Vertex unchanged}
        $\vec{\mathbf{P}}^*_{i+1} \leftarrow \vec{\mathbf{P}}_i$
$m^* = m$
```

2.3.7 B-spline surface fitting of grid points

The functional design of a foil mean camber line or surface involves finding the shape of a foil satisfying a set of physical constraints. Before introducing how one includes these physical constraints into a design problem, however, a brief overview of fitting a B-spline surface to a grid of data will provide an introduction to the traditional method of fitting a B-spline (through interpolation or approximation) to a set a data.

A set of grid points $\{\vec{Q}_{ij} : i = 0, \cdots, m_p - 1;\ j = 0, \cdots, n_p - 1\}$ is initially given. The (position) approximation of the point set using a B-spline surface in form of Equation 2.9 in the least-squares sense (called B-spline least-squares fitting [14]) involves finding the control vertices $\{\vec{S}_{ij} : i = 0, \cdots, m - 1;\ j = 0, \cdots, n - 1\}$ such that, with predetermined orders k, l and knot vectors $\mathbf{U}$ and $\mathbf{V}$, the following total position approximation error E_P is minimized:

$$E_{\vec{Q}} = \sum_{i_p=0}^{m_p-1} \sum_{j_p=0}^{n_p-1} \left[\vec{S}(u_{i_p}, v_{j_p}) - \vec{Q}_{i_p j_p}\right]^2 = min. \tag{2.12}$$

where u_{i_p} and v_{j_p} are the parametric coordinates assigned to each grid point.

That is to say,

$$E_{\vec{Q}} = \sum_{i_p=0}^{m_p-1} \sum_{j_p=0}^{n_p-1} \left[\sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \vec{S}_{ij} B_i^k(u_{i_p}) \bar{B}_j^l(v_{j_p}) - \vec{Q}_{i_p j_p}\right]^2 = min. \tag{2.13}$$

This minimization problem can be converted into the following linear equation system(s):

$$\frac{\partial E_{\vec{Q}}}{\partial \vec{S}_{\bar{i}\bar{j}}} = 0 \qquad (\bar{i} = 0, 1, \cdots, m-1; \ \bar{j} = 0, 1, \cdots, n-1) \tag{2.14}$$

That is to say:

$$\begin{aligned} \sum_{i_p=0}^{m_p-1} \sum_{j_p=0}^{n_p-1} a_{\bar{i}i_p} \bar{a}_{\bar{j}j_p} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} a_{ii_p} \bar{a}_{jj_p} \vec{S}_{i_q j_q} \quad &= \quad \sum_{i_p=0}^{m_p-1} \sum_{j_p=0}^{n_p-1} a_{\bar{i}i_p} \bar{a}_{\bar{j}j_p} \vec{Q}_{i_p j_p} \\ &\quad (\bar{i} = 0, 1, \cdots, m-1; \ \bar{j} = 0, 1, \cdots, n-1) \end{aligned} \tag{2.15}$$

where, $a_{ii_p} = B_i^m(u_{i_p})$ and $\bar{a}_{jj_p} = \bar{B}_j^n(v_{j_p})$.

Equation 2.15 is actually a vector-valued banded matrix equation (i.e., many linear equation systems) which can be solved row- and column-wise or converted into one vector-value banded linear equation system by arranging the control mesh into a control vertex array. In both approaches the equation system can be efficiently solved by using the fact that the coordinates X, Y, and Z of $\vec{S}(u, v)$ and $\vec{Q}_{ij}$ are independent of each other permitting the reduction of the larger (three times as large in row and column) system to three smaller linear equation systems with a common coefficient matrix.

The order (k, l) and number of the control vertices (m, n) of the surface $\vec{S}(u, v)$ is usually specified by the user. The bicubic case (i.e., $k = l = 4$) is the most frequently used, since this is the lowest degree which can provide curvature continuous surfaces demanded by most applications.

The knot vectors **U** and **V** are usually automatically computed by the approximation method. The most frequently used approach for least-squares fitting is the (quasi-)uniform knots [14] which are normalized as follows:

$$\mathbf{U} = \left\{\underbrace{0,0,\cdots,0}_{k},\frac{1}{m-k+1},\frac{2}{m-k+1},\cdots,\frac{m-k}{m-k+1},\underbrace{1,1,\cdots,1}_{k}\right\} \quad (2.16)$$

$$\mathbf{V} = \left\{\underbrace{0,0,\cdots,0}_{l},\frac{1}{n-l+1},\frac{2}{n-l+1},\cdots,\frac{n-l}{n-l+1},\underbrace{1,1,\cdots,1}_{l}\right\} \quad (2.17)$$

There are two frequently used approaches to choose the parametric values $\{(u_{i_p}, v_{j_p}) : i_p = 0,1,\cdots,m_p-1;\ j_p = 0,1,\cdots,n_p-1\}$ corresponding to the given grid points. One is the uniform approach. In which case, the parametric points are uniformly distributed in the parametric space. They are chosen as follows:

$$u_{i_p} = \frac{i_p}{m_p-1} \quad (i_p = 0,1,\cdots,m_p-1); \qquad v_{j_p} = \frac{j_p}{n_p-1} \quad (i_p = 0,1,\cdots,n_p-1) \quad (2.18)$$

Another usually better approach is to choose the parametric values by using the average chord-lengths. For example,

$$u_0 = 0; \qquad u_{i_p} = \frac{\sum_{k=1}^{i_p}\sum_{j_p=0}^{n_p-1}|\vec{Q}_{kj_p} - \vec{Q}_{k-1,j_p}|}{\sum_{k=1}^{m_p-1}\sum_{j_p=0}^{n_p-1}|\vec{Q}_{kj_p} - \vec{Q}_{k-1,j_p}|} \quad (i_p > 0) \quad (2.19)$$

For interpolation of the data points, the number of degrees of freedom in the B-spline surface must be equal to the number of equations. If $m = m_p$ and $n = n_p$, an exact solution can be found and the resulting B-spline surface will interpolate the set of data points. Mathematically, for interpolation, the objective function representing the error in Equation 2.13 becomes:

$$E_{\vec{Q}} = \sum_{i_p=0}^{m_p-1}\sum_{j_p=0}^{n_p-1}\left[\sum_{i=0}^{m_p-1}\sum_{j=0}^{n_p-1}\vec{S}_{ij}B_i^k(u_{i_p})\bar{B}_j^l(v_{j_p}) - \vec{Q}_{i_p,j_p}\right]^2 = 0 \quad (2.20)$$

The technique presented here is used to fit a B-spline surface to a set of data.

Fitting a curve is the simpler case where $n_p = 1$, reducing the complexity of the process but the same fundamental approach still holds. In many design schemes, the design process starts with a set a data roughly representing the desired shape and then fitting a B-spline curve or surface to it. To achieve the final shape, the control vertices of the B-spline geometry are tweaked using performance related criteria.

2.4 Design process of lifting surfaces

The design of lifting surfaces can be divided into two general methods: optimization and inverse design. The first method, optimization, involves the assumption of an initial shape which then is analyzed, by using either computational analysis or experimental techniques, and iteratively adjusted until the observed loading is optimum. The inverse design method, on the other hand, consists of initially prescribing a desired loading over the lifting shape and then modifying the shape to achieve this loading without violating any physical constraints. This method involves prescribing the loading (pressure distribution) over the planform of a wing, foil, or propeller blade from the start. Since the initial shape of the lifting surface is a guess (usually based upon past experience), the flow generating the desired pressure distribution may not be physically possible. The design process then becomes a problem of finding the shape which carries this lift distribution without violating the boundary conditions. Of the two methods, typically the inverse method of design requires less computing time [13, 19] and is, therefore, more desirable. To introduce the inverse design method, the Kutta-Joukowski theorem will first be presented, giving an equation for the computation of the lift force experienced by a single vortex in a flow. The theorem will then be expanded to represent a series a vortices from which a qualitative explanation of how a mean camber lifting surface can be found through the inverse design method.

The Kutta-Joukowski theorem relates the lift L experienced by a vortex to its strength Γ, the free stream velocity of the flow V_∞, and the fluid density ρ

$$L = \rho V_\infty \Gamma. \tag{2.21}$$

The lift acts perpendicular to the direction of the flow had the vortex not been present [3, 25].

Through potential flow theory, it is possible to represent the flow about a lifting body by replacing the body with a distribution of vorticity. In conjunction with the Kutta-Joukowski theorem, it then becomes possible to relate the lift generated by a lifting surface to a corresponding distribution of vorticity over the surface which models the fluid flow. The total vorticity, or circulation, about the lifting surface can be inserted directly into the Kutta-Joukowski theorem to obtain a calculation of the lift acting on the surface.

Instead of analyzing the lifting surface and its surrounding flow for the amount of lift produced, the inverse design method goes the other way. For the desired distribution of lift over a given planform, the necessary vorticity distribution is computed and the design method searches for the shape which the mean camber line or surface must have in order to satisfy both the kinematic boundary condition and the Kutta condition. To accomplish this, potential flow theory is used to calculate the velocity over the initial shape from the free stream velocity and the prescribed distribution of vorticity. The flow is then checked to see if it satisfies the kinematic boundary condition. Most likely there will be some degree of violation and the shape is altered in such a way as to better satisfy this physical constraint. The Kutta condition can be met from the start if the prescribed vorticity vanishes at the sharp trailing edge of the foil. The process continues iteratively until the physical constraints are satisfied and a shape is found which will carry the prescribed loading.

2.5 Literature review

Functional design of shapes is not a new concept to the CAD/CAGD and engineering communities. The method of geometric representation and how to apply the physical constraints to the design process, however, has been evolving over the years. The inverse design method described is one example of functional design which has been used successfully for the design of airfoil shapes and mean camber surfaces of wings and propeller blades. In this section, a survey of some of the techniques developed by other researchers in the field will be introduced. First, two different methods of airfoil design will be presented. Next, a novel method of geometric representation and functional design involving the description of a propeller's geometry in terms of partial differential equations will be discussed. Finally, the design of propeller blades using B-spline geometry will be explained from which the motivation of this work will become apparent.

The first example of airfoil design comes from Giles and Drela [19], which actually uses a mixed inverse design method for arriving at a new airfoil shape. Part of the geometry of a foil is initially prescribed while the pressure distribution is specified along the undefined section (which is most of the upper surface). Through a discrete volume formulation of the Euler equations governing the flow, the flow is analyzed and the shape of the undefined suction edge is found such that it satisfies the kinematic boundary condition as well as a condition of geometric continuity along the top surface of the airfoil.

The next example of airfoil design, from Eppler [13], uses a panel method to solve for the flow. As previously described, an inviscid, irrotational flow can be modeled with a distribution of singularities outside of the fluid domain. This method involves modeling the airfoil as a polygon of line segments which each have an associated dipole. The strengths of the dipoles are specified according to the desired loading. Next, a Newton method is used to adjust the shape of the foil, minimizing the error in satisfying the kinematic boundary condition. When the solution converges, an airfoil shape which will carry the prescribed load is found.

More recently, a Partial Differential Equation (PDE) method has been developed by Bloor, Wilson and their students in which the geometry of the functional shape is defined terms of a PDE with prescribed geometric boundary conditions [7, 6, 5, 12, 23]. A surface is the solution of an elliptic or biharmonic partial differential equation with suitably chosen boundary conditions. Other examples include the methods introduced in [26, 27, 40]. As an example, for the functional design of a propeller blade, an initial blade shape is first assumed in terms of a PDE geometry. Next, the shape is evaluated and its performance analyzed through the use of a panel method. From the analysis, form parameters in the PDE description of the geometry are altered to optimize the blade's geometry. Despite the fact that the blade shape is iteratively analyzed using a panel method, the design process is expected to be fast since the cost of evaluation of the PDE geometry is negligible compared to the computational cost of the analysis and the number of parameters (degrees of freedom) in the PDE can be kept to a minimum while still providing sufficient flexibility to represent complex shapes [12]. An overview of the methods for the systematic variations of ship hull form characteristics and for their evaluation from a hydrodynamic point of view is given in [26]. The form parameters used in [26] include local parameters (data points and derivatives at these points), regional form parameters (characteristics of forebody, midbody, afterbody, etc.), and global form parameters (displacement, centroids, principal dimensions, etc.). The form variation problems are converted to nonlinear optimization problems with geometric and physical constraints such as area constraints [27] and volume constraints [40].

A very popular method of shape representation involves the use of B-spline functions. With the standardization of B-splines to represent shapes in CAD/CAM systems and to exchange such shapes between systems, designers have started to incorporate B-spline geometries directly into their functional design methods. An example of this is the method developed by Kerwin *et al.* [22] in which B-spline surfaces are incorporated directly into the design process of marine propulsors. In their method, the mean camber surface of a propulsor blade is described in terms of a B-spline surface. The control vertices represent the degrees of freedom for adjusting the shape which are adjusted so as to satisfy the kinematic boundary condition over the surface. Since this method of functional design is the basis and motivation behind this thesis, a more detailed explanation of the method is warranted.

Kerwin *et al.*'s design process [22] begins by assuming an initial propulsor geometry, which may be derived from experience or from a former propulsor shape. Next, a desired loading is prescribed. With an initial shape and target loading, a design loop is entered through which the shape is modified. The first step of the loop is the evaluation of the velocity field of the ship's wake (in which the propulsor will operate) through the use of an axisymmetric, Reynolds Averaged Navier-Stokes analysis program. From this velocity field, along with additional velocity induced by the prescribed loading distribution, the total velocity at the mean camber surface is evaluated. Since the velocity at these points may not satisfy the kinematic boundary condition, the magnitude of the violation of this physical constraint must be evaluated over the surface. If the violation is greater than a maximum allowable tolerance, the control vertices of the geometry are iteratively adjusted (one at a time) by the computer according to a Newton-Rhapson scheme to find the shape which minimizes the normal component of the total fluid velocity at a grid of points (in a least squares sense) [22]. Since the geometry of the blade has changed, the process is repeated to find the new velocity field and the kinematic boundary condition is checked again. When the fluid flux through the surface is found to be below an allowable threshold, the solution is considered to have converged. The final shape represents the mean camber surface of a propulsor producing the desired thrust. This design process is illustrated in Figure 2-7.

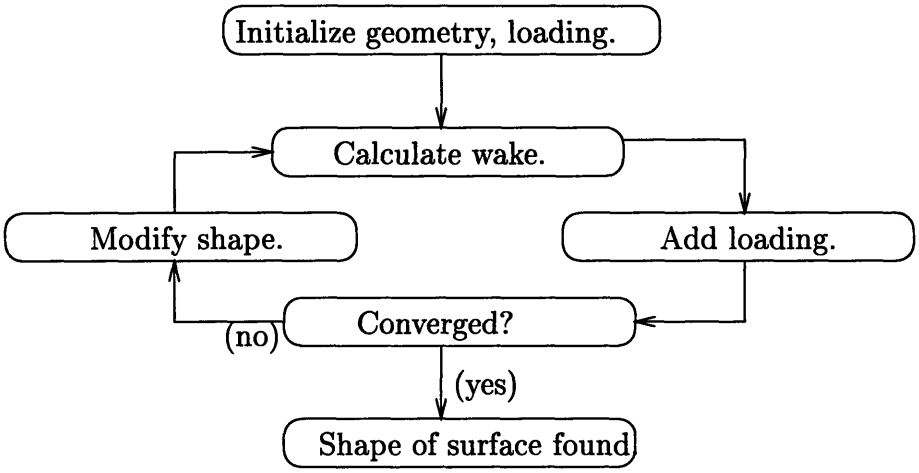


Figure 2-7: Design loop for the functional design of propeller blades, as described by Kerwin *et al.*[22].

Chapter 3

Shape modification through approximation of a grid of velocity vectors

The first approach to functional surface design involves examining the "Modify shape" step in the propulsor design loop shown in Figure 2-7. This step begins with an initial blade geometry and a velocity field evaluated at a grid of points on the surface of the blade. To arrive at this step, the convergence check had to indicate that the current geometry violates the kinematic boundary condition beyond an allowable threshold. This section presents a method to improve the shape of the initial blade to better satisfy the kinematic boundary condition based upon the approximation of a grid of velocity vectors and the current blade geometry. Several examples are then given utilizing this shape modification approach in the design process developed by Kerwin *et al.* [22]. A summary of the work presented in this chapter has been published in the *Computer Aided Design Journal* by Ye, Jackson, and Patrikalakis [44].

3.1 Formulation of shape modification

3.1.1 Statement of problem

Let us first consider the kinematic boundary condition given in Equation 2.4 on the propeller blade's mean camber surface. There are two possible approaches to solving for a surface producing a desired distribution of lift while satisfying this physical constraint. The first method satisfies the boundary condition at every point of the desired surface whereas the other is concerned with satisfying it at only a discrete set of points on the surface. The former way involves functional variation which is not convenient due to its high complexity. The latter approach, however, requires that the boundary condition be satisfied on a grid of points, allowing conventional fitting techniques (as described in Section 2.3.7) to be used. In CAD and CAGD, the latter way is generally preferred. For this reason, this thesis deals only with the discretized form of the physical constraints. The problem of modifying the design of a propeller blade can thus be formulated as follows:

Given a grid of points $\{\vec{Q}_{i_p j_p} : i_p = 0, \cdots, m_p - 1;\ j_p = 0, \cdots, n_p - 1\}$ *sampled from the original surface as well as the velocity vectors* $\{\vec{V}_{i_p j_p}\}$ *at these grid points (see Figure 3-1), find a new surface* $\vec{S}(u, v)$ *which satisfies (in the least-squares sense) the kinematic boundary condition (Equation 2.4) at these grid points.*

Mathematically, the discretized form of the shape modification problem can be represented as follows:

$$\begin{aligned} \vec{S}(u_{i_p}, v_{j_p}) &= \vec{Q}_{i_p j_p} \qquad (i_p = 0, \cdots, m_p - 1; j_p = 0, \cdots, n_p - 1) \quad (3.1) \\ \hat{n}(u_{i_p}, v_{j_p}) \cdot \vec{V}_{i_p j_p} &= 0 \qquad (i_p = 0, \cdots, m_p - 1; j_p = 0, \cdots, n_p - 1) \quad (3.2) \\ where \qquad & \hat{n}(u_{i_p}, v_{j_p}) = \frac{\vec{S}_u(u_{i_p}, v_{j_p}) \times \vec{S}_v(u_{i_p}, v_{j_p})}{|\vec{S}_u(u_{i_p}, v_{j_p}) \times \vec{S}_v(u_{i_p}, v_{j_p})|} \end{aligned}$$

where (u_{i_p}, v_{j_p}) are the parametric values corresponding to the point $\vec{Q}_{i_p j_p}$ and $\vec{V}_{i_p j_p}$ is the flow velocity at the sampled point on the initial surface, and subscripts u, v denote partial derivatives. Notice that Equation 3.1 implies simply the fitting of a

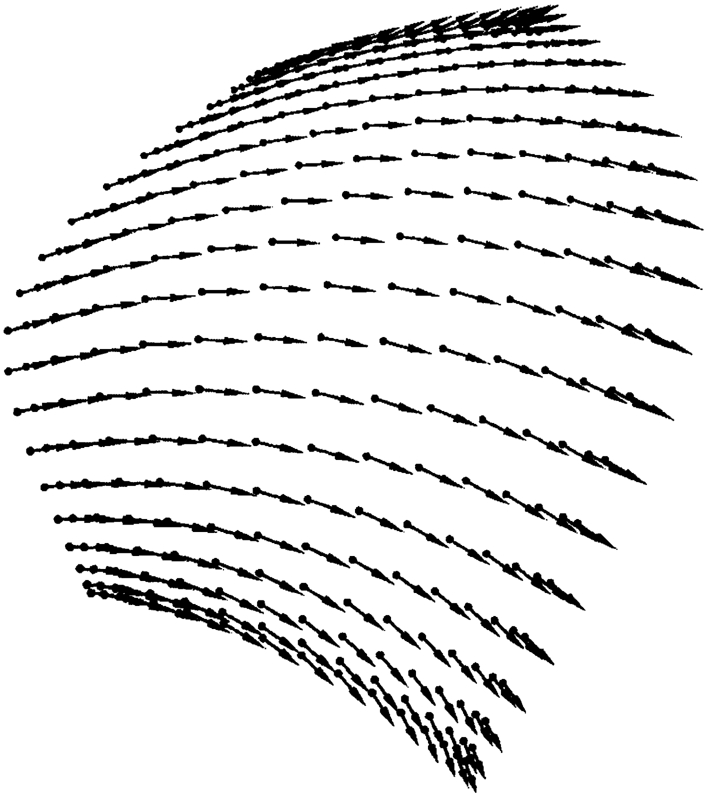


Figure 3-1: The grid points sampled over a blade surface with the corresponding flow velocity vectors shown at these points.

surface to a grid of points, as described in Section 2.3.7. Equation 3.2 introduces the physics, imposing the kinematic boundary condition on the solution. Taken together, Equation System 3.1–3.2 is a *non-linear vector equation system.* This can be clearly seen if we assume the target surface $\vec{S}$ to be in integral B-spline form. In this case, the surface's normal $\hat{n}$ in Equation 3.2 is nonlinearly related to the positions of the control vertices of $\vec{S}$, yielding the nonlinear equation system:

$$\left[\vec{S}_u(u_{i_p}, v_{j_p}) \times \vec{S}_v(u_{i_p}, v_{j_p})\right] \cdot \vec{V}_{i_p j_p} = 0 \qquad (i_p = 0, \cdots, m_p - 1; j_p = 0, \cdots, n_p - 1) \quad (3.3)$$

Solving a non-linear equation system is generally very time-consuming. Therefore, non-linear equation systems are often solved by local methods based on linearization which introduces approximation error and makes the solution imprecise. However, the non-linear Equation System 3.1–3.3 can be converted to a linear equation system by introducing auxiliary variables. The solution of this linear system is a subset of the solution set of the initial non-linear equation system. This conversion has the advantages that it reduces the complexity of the problem, saves computation time, and increases the precision and reliability of the solution.

3.1.2 Conversion of hydrodynamic constraints to linear form

Equations 3.1 and 3.3 can be converted to the following constraints by assuming that a grid of desirable surface normal vectors $\hat{n}_{i_pj_p}$ can be determine a priori:

Find a surface $\vec{S}(u,v)$, such that $\vec{S}(u,v)$ fits the given position vectors $\{\vec{Q}_{i_pj_p} : (i_p = 0, \cdots, m_p - 1;\ j_p = 0, \cdots, n_p - 1)\}$ of a grid of points and prescribed normal vectors $\{\hat{n}_{i_pj_p} : (i_p = 0, \cdots, m_p - 1; j_p = 0, \cdots, n_p - 1)\}$ which are perpendicular to the corresponding velocity vectors $\{\vec{V}_{i_pj_p} : (i_p = 0, \cdots, m_p - 1; j_p = 0, \cdots, n_p - 1)\}$ at the grid points (see Figure 3-4).

Mathematically, the above constraints can be represented as follows:

$$\begin{aligned} \vec{S}(u_{i_p}, v_{j_p}) &= \vec{Q}_{i_pj_p} && (3.4) \\ & \qquad (i_p = 0, \cdots, m_p - 1;\ j_p = 0, \cdots, n_p - 1) \\ \vec{S}_u(u_{i_p}, v_{j_p}) \times \vec{S}_v(u_{i_p}, v_{j_p}) &\parallel \hat{n}_{i_pj_p} && (3.5) \\ & \qquad (i_p = 0, \cdots, m_p - 1;\ j_p = 0, \cdots, n_p - 1) \end{aligned}$$

where, $\hat{n}_{i_pj_p}$ is the desired surface normal vector satisfying the kinematic boundary condition at the grid point $\vec{Q}_{i_pj_p}$.

Equation System 3.4–3.5 is again a *non-linear vector equation system.* Fortunately, Equation 3.5 can be converted to two linear equations. To see this, as shown in Figure 3-2a, the condition that the cross product of two vectors $\vec{A}$ and $\vec{B}$ is parallel to a third vector $\vec{C}$ requires that $\vec{A}$ and $\vec{B}$ lie in the plane normal to $\vec{C}$. Therefore, the inner products of $\vec{A}$ with $\vec{C}$ and $\vec{B}$ with $\vec{C}$ must be zero. Hence, Equation System 3.4–3.5 is equivalent to the following *linear equation system:*

$$\vec{S}(u_{i_p}, v_{j_p}) = \vec{Q}_{i_pj_p} \qquad (i_p = 0, \cdots, m_p - 1;\ j_p = 0, \cdots, n_p - 1) \quad (3.6)$$

$$\vec{S}_u(u_{i_p}, v_{j_p}) \cdot \hat{n}_{i_pj_p} = 0 \qquad (i_p = 0, \cdots, m_p - 1;\ j_p = 0, \cdots, n_p - 1) \tag{3.7}$$

$$\vec{S}_v(u_{i_p}, v_{j_p}) \cdot \hat{n}_{i_pj_p} = 0 \qquad (i_p = 0, \cdots, m_p - 1;\ j_p = 0, \cdots, n_p - 1) \tag{3.8}$$

Notice that the linear equation system 3.6–3.8 is not equivalent to the non-linear equation system 3.1–3.3. The number of equations in the linear equation system at each grid point of the surface increases from four in the original non-linear equation system to five. Selection of $\hat{n}_{i_pj_p}$ represents one degree of freedom in the non-linear equation system. By imposing a constraint on $\hat{n}(u_{i_p}, v_{j_p})$ of the surface $\vec{S}(u, v)$, a subset of all the possible solutions of the non-linear equation system can be found from the linear equation system.

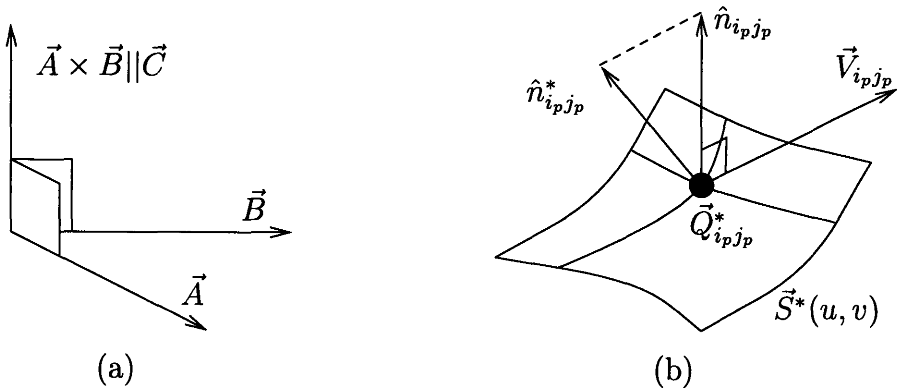


Figure 3-2: (a): Transform the non-linear vector parallelization problem to the linear zero inner product problem: $\vec{A} \times B \parallel \vec{C}$ is equivalent to the requirement that $\vec{A} \cdot \vec{C} = \vec{B} \cdot \vec{C} = 0$; (b): The normal vector $\hat{n}_{i_pj_p}$ of the surface at $\vec{Q}_{i_pj_p}$ is taken to be the projection of the normal $\hat{n}^*_{i_pj_p}$ of the initial surface $\vec{S}^*(u, v)$ into the plane perpendicular to the velocity vector $\vec{V}_{i_pj_p}$.

3.1.3 Specification of the target normal vectors

To solve the linear equation system 3.6–3.8, the normal vectors $\hat{n}_{i_pj_p}$ $(i_p = 0, \cdots, m_p - 1,\ j_p = 0, \cdots, n_p - 1)$ at the given grid points $\vec{Q}_{i_pj_p}$ must be available. They may be specified by the user or determined automatically by using the method presented here. With the belief that the initial design is a good approximation to the final design after modifying the shape, the following steps can be used to determine these normal vectors:

Determine the desired unit normal vector $\hat{n}_{i_pj_p}$ *of the new surface at* $\vec{Q}_{i_pj_p}$ *such that* $\hat{n}_{i_pj_p}$ *is the normalized projection of the original surface normal* $\hat{n}^*_{i_pj_p}$ *into the plane perpendicular to the velocity vector* $\vec{V}_{i_pj_p}$ *(see Figure 3-2b). This is equivalent to determining the unit normal to the plane defined by the velocity vector* $\vec{V}_{i_pj_p}$ *and a vector perpendicular to initial normal and velocity vector* $\hat{n}^*_{i_pj_p} \times \vec{V}_{i_pj_p}$ *:*

$$\hat{n}_{i_pj_p} = \frac{\vec{V}_{i_pj_p} \times (\hat{n}^*_{i_pj_p} \times \vec{V}_{i_pj_p})}{|\vec{V}_{i_pj_p} \times (\hat{n}^*_{i_pj_p} \times \vec{V}_{i_pj_p})|}$$

Recognizing the above equation as the vector triple product, the equation can be simplified as a ratio of scalar products:

$$\hat{n}_{i_pj_p} = \frac{(\vec{V}_{i_pj_p} \cdot \vec{V}_{i_pj_p})\hat{n}^*_{i_pj_p} - (\vec{V}_{i_pj_p} \cdot \hat{n}^*_{i_pj_p})\vec{V}_{i_pj_p}}{|(\vec{V}_{i_pj_p} \cdot \vec{V}_{i_pj_p})\hat{n}^*_{i_pj_p} - (\vec{V}_{i_pj_p} \cdot \hat{n}^*_{i_pj_p})\vec{V}_{i_pj_p}|} \tag{3.9}$$

Figure 3-3 shows the unit normal vectors of the initial blade surface at the given grid points corresponding to the velocity field in Figure 3-1 . Figure 3-4 shows the desired normal vectors for the modified at the grid points as computed by Equation 3.9.

3.2 B-spline surface fitting with normal vector constraints

3.2.1 B-spline surface approximation vs. interpolation

The linear equation system 3.6 to 3.8 for a B-spline surface $\vec{S}(u,v)$ can be solved in the sense of either strict interpolation or approximation. Although there are many applications which require interpolation [14], for this problem we choose the approximation (least-squares) fitting method based on the following considerations:

1. In engineering applications, frequently the measured/computed data themselves are only approximations;

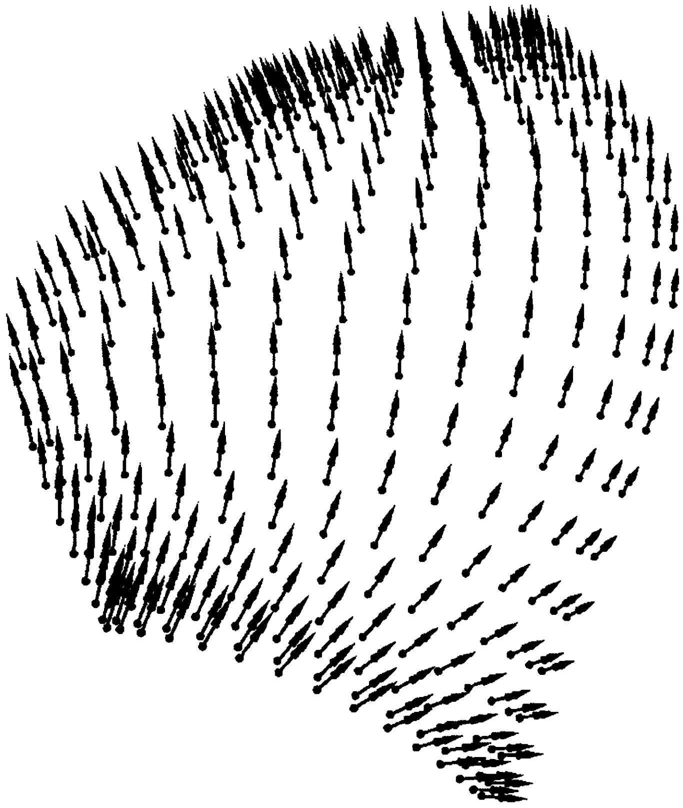


Figure 3-3: The unit normal vectors of the initial blade surface $\{\hat{n}^*_{i_p j_p} : i_p = 0, \cdots, m_p - 1; j_p = 0, \cdots, n_p - 1\}$ at the grid points $\{\vec{Q}_{i_p j_p} : i_p = 0, \cdots, m_p - 1; j_p = 0, \cdots, n_p - 1\}$. The surface is generated by least-squares fitting of the grid points in Figure 3-1.

2. Practical experience has shown that B-spline approximation provides more reasonable and fair results than interpolation;

3. There exists a practical difficulty in solving the linear equation system 3.6–3.8 by interpolation. The difficulty becomes clear when we analyze the number of equations in the system. Equation 3.6 has $m_p n_p$ vector equations, i.e., $3m_p n_p$ equations. Equations 3.7 and 3.8 each provide $m_p n_p$ equations. So the total number of equations is $5m_p n_p$. If we assume that the number of the control vertices of the B-spline surface $\vec{S}$ is mn, then we have $3mn$ unknowns (in three dimensions x, y, z). To make the equation system balanced and therefore achieve

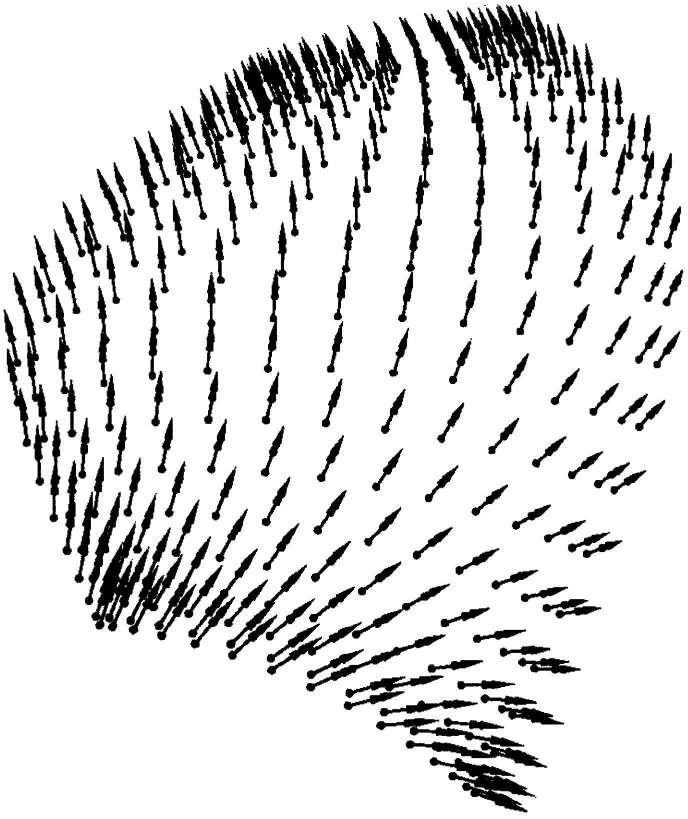


Figure 3-4: The prescribed unit normal vectors of the blade surface at the grid points in Figure 3-1. They are generated by the method shown in Figure 3-2(b).

interpolation, the number of equations and unknowns should be the same, i.e.,

$$3mn = 5m_p n_p \tag{3.10}$$

Equation 3.10 is not generally solvable for arbitrary integer values of m, n, m_p, and n_p. By choosing to use least squares fitting, we can approximate a new blade shape even when Equation 3.10 is not satisfied.

3.2.2 B-spline surface least squares fitting with normal vector constraints

Similar to the B-spline least squares solution of grid points (see Section 2.3.7), the least-squares solution of the equation system 3.6–3.8 can be mathematically formulated as the problem of minimizing an error function E defined as the weighted combination of the errors in approximating the position data and the normal constraints:

$$E = E_{\vec{Q}} + wE_{\hat{n}} = min. \tag{3.11}$$

$E_{\vec{Q}}$ is the position approximation error previously defined by 2.13 for the regular approximation of a grid of data. $E_{\hat{n}}$ is the normal vector approximation error accounting for the physical constraints. The influence between the two is determined by the weighting factor w.

The normal approximation error $E_{\hat{n}}$ is defined as follows:

$$\begin{aligned} E_{\hat{n}} &= \sum_{i_p=0}^{m_p-1}\sum_{j_p=0}^{n_p-1}\left\{\left[\vec{S}_u(u_{i_p}, v_{j_p})\cdot\hat{n}_{i_pj_p}\right]^2 + \left[\vec{S}_v(u_{i_p}, v_{j_p})\cdot\hat{n}_{i_pj_p}\right]^2\right\} \\ &= \sum_{i_p=0}^{m_p-1}\sum_{j_p=0}^{n_p-1}\left\{\left[\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}a'_{ii_p}\bar{a}_{jj_p}(\vec{S}_{ij}\cdot\hat{n}_{i_pj_p})\right]^2 + \left[\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}a_{ii_p}\bar{a}'_{jj_p}(\vec{S}_{ij}\cdot\hat{n}_{i_pj_p})\right]^2\right\}, \end{aligned}$$

where $a'_{ii_p} = B_i'^k(u_{i_p})$ and $\bar{a}'_{jj_p} = \bar{B}_j'^l(v_{j_p})$.

Similar to Equation 2.13, the minimization problem posed by Equation 3.11 can be converted to the following linear equation system:

$$\frac{\partial E}{\partial\vec{Q}_{\bar{i}\bar{j}}} = 0 \qquad (\bar{i} = 0, 1, \cdots, m-1;\ \bar{j} = 0, 1, \cdots, n-1) \tag{3.12}$$

That is to say,

$$\begin{aligned} &\sum_{i_p=0}^{m_p-1}\sum_{j_p=0}^{n_p-1}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\Big\{a_{\bar{i}i_p}\bar{a}_{\bar{j}j_p}a_{ii_p}\bar{a}_{jj_p}\vec{S}_{ij} + \\ &\qquad w(a'_{\bar{i}i_p}\bar{a}_{\bar{j}j_p}a'_{ii_p}\bar{a}_{jj_p} + a_{\bar{i}i_p}\bar{a}'_{\bar{j}j_p}a_{ii_p}\bar{a}'_{jj_p})(\hat{n}_{i_pj_p}\cdot\vec{S}_{ij})\hat{n}_{i_pj_p}\Big\} \\ &= \sum_{i_p=0}^{m_p-1}\sum_{j_p=0}^{n_p-1}a_{\bar{i}i_p}\bar{a}_{\bar{j}j_p}\vec{Q}_{i_pj_p} \quad (\bar{i} = 0, 1, \cdots, m-1;\ \bar{j} = 0, 1, \cdots, n-1) \end{aligned} \tag{3.13}$$

The weight w has a great affect on the shape of the final surface $\vec{S}(u, v)$. Letting $w = 0$ results in only position approximation, as was presented in Section 2.3.7. By increasing w, the prescribed normal vectors for the new surface at the grid points contribute more to Equation 3.11, and therefore contribute more to the shape of the surface. The weight w should be chosen carefully. The decision about how much weight to apply to the prescribed normal vectors is entirely data dependent. With greater weight, the deviation from the original design can significantly increase. Depending upon the application, the designer must decide how much of the shape will be dictated by the prescribed normals.

Notice that in Equation 3.13, because of the presence of the term $\hat{n}_{i_p j_p} \cdot \vec{S}_{ij}$, the x, y, and z components of the control vertices *can not* be treated independently as in Equation 2.15, a unique feature of the fitting process with normal constraints. In matrix form, Equation 3.13 can be expressed as follows.

First, let the 3 × 3 submatrix $\mathbf{M}^{ij}_{\bar{i}\bar{j}}$ be defined as:

$$\begin{aligned}\mathbf{M}^{ij}_{\bar{i}\bar{j}} = &\left\{ \sum_{i_p=0}^{m_p-1} \sum_{j_p=0}^{n_p-1} a_{\bar{i}i_p}\bar{a}_{\bar{j}j_p}a_{ii_p}\bar{a}_{jj_p} \right\} \mathbf{I}_{3\times 3} + \\ &w \left\{ \sum_{i_p=0}^{m_p-1} \sum_{j_p=0}^{n_p-1} \left(a'_{\bar{i}i_p}\bar{a}_{\bar{j}j_p}a'_{ii_p}\bar{a}_{jj_p} + a_{\bar{i}i_p}\bar{a}'_{\bar{j}j_p}a_{ii_p}\bar{a}'_{jj_p} \right) \right\} \hat{n}_{i_p j_p} \cdot \hat{n}^T_{i_p j_p} \end{aligned} \tag{3.14}$$

where $\mathbf{I}_{3\times 3}$ is the 3 × 3 unit matrix.

With the definition of the submatrix $\mathbf{M}^{ij}_{\bar{i}\bar{j}}$ in Equation 3.14, the linear Equation System 3.13 can be rewritten more compactly as:

$$\begin{aligned}\sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \mathbf{M}^{ij}_{\bar{i}\bar{j}} \vec{S}_{ij} &= \sum_{i_p=0}^{m_p-1} \sum_{j_p=0}^{n_p-1} a_{\bar{i}i_p}\bar{a}_{\bar{j}j_p}\vec{Q}_{i_p j_p} \\ &(\bar{i} = 0, 1, \cdots, m-1;\ \bar{j} = 0, 1, \cdots, n-1) \end{aligned} \tag{3.15}$$

The coefficient matrix of Equation 3.15 is a sparse matrix. In fact, it is banded. The solvability of the equation system Equation 3.15, i.e., the non-singularity of the coefficient matrix is not easy to prove theoretically. However, according to our experience, if the number of the node points is chosen not to exceed the number of the grid points given in both of the $\hat{u}$ and $\hat{v}$ directions, i,e., $m \leq m_p$ and $n \leq n_p$, then the coefficient matrix is most likely full ranked and well-conditioned resulting in a fair surface. When more node points are used ($m > m_p$ or $n > n_p$), the additional degrees of freedom may introduce oscillations in the surface. Considering that $m \leq m_p$ and $n \leq n_p$ are required in applications, we can conclude that the linear equation system provides reliable solutions in practice. Due to the local support property of the B-spline basis functions [14], the matrix is sparse. Therefore, to solve the linear equation system efficiently we use algorithms specifically designed to take advantage of the matrix's sparsity, solving the system quickly and efficiently [29, 18].

In some applications, some points are more important than others in terms of approximation error. For example, in the functional design of blade surfaces, it is sometimes desirable to maintain the new leading edge of the modified shape as close as possible to the original (see Figure 3-6). In this case, additional weights should be attached to the points representing the leading edge and the position approximation error in Equation 2.13 should be defined as:

$$E_{\vec{Q}} = \sum_{i_p=0}^{m_p-1} \sum_{j_p=0}^{n_p-1} w_{i_p j_p} \left[\sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \vec{S}_{ij} B_i^k(u_{i_p}) \bar{B}_j^l(v_{j_p}) - \vec{Q}_{i_p j_p} \right]^2 = min. \tag{3.16}$$

where, $w_{ij} > 0$ is the weight attached to the point $\vec{Q}_{i_p j_p}$. Generally, the greater the weight attached to a point, the closer the new B-spline surface will be to that point. Therefore, greater weights can be attached to the grid points which need to be closely approximated.

3.3 Implementation of approximation method

This section presents some examples and discussion of the functional modification of propeller blade surfaces. To facilitate testing of this functional design method, we incorporated it with the propeller design program PBD described in Section 2.5 and [22]. The surfaces are displayed by PRAXITELES, a geometric modeling and interrogation system developed in the MIT Design Laboratory [1].

To implement the functional design process described in this thesis, the shape modification step in Figure 2-7 was replaced. In the original version of PBD used for this thesis, the modification of the blade shape involved successively adjusting the position of each control vertex of the B-spline net in order to minimize the error in satisfying Equation 2.4. After several iterations of adjusting the control vertices, the velocities at the node points on the new blade surface were computed and the satisfaction of the kinematic boundary condition checked. The design loop given in Figure 2-7 was then repeated until the shape finally converges (i.e. no further improvement in satisfying Equation 2.4 is required). In order to test our method, the iterative process of moving individual control vertices of the B-spline net was replaced with the functional surface fitting method described in this section to modify the blade shape. With this method, a new B-spline net is found by solving one system of linear equations in one step rather than sequentially adjusting the control vertices of the initial surface individually.

The first test of this functional design method consists of fitting surfaces to the grid of points on the initial blade surface (Figure 3-6) for which PBD has computed the corresponding flow velocities (Figure 3-1) for different weighting schemes. For this case, the initial blade is approximately 0.8 m high with a chord length of 0.3 m. The maximum velocity of the flow over the blade is 3.75 m/s. From the surface approximating the grid points and the velocity vectors at these points, the prescribed normal vectors are computed, as shown in Figure 3-4. The reduction in the error associated with the physical constraint before and after shape modification can be seen in Figure 3-5.

The errors associated with the new surfaces approximating the grid points and velocity vectors are shown in Tables 3.1 and 3.2. The average and maximum errors associated with approximating position ($|\vec{S}(u_i^*, v_j^*) - \vec{Q}_{ij}|$) and the boundary condition ($|(\vec{S}_u(u_i^*, v_j^*) \times \vec{S}_v(u_i^*, v_j^*)) \cdot \vec{V}_{ij}|$) over all grid points are listed for increased weighting w of the prescribed normals. The average and maximum angles (in degrees) between the tangent plane of the functional surface and the velocity vector ($\angle(T_S(u_i^*, v_j^*), \vec{V}_{ij})$) are also given in the last column to provide a physical understanding of the effect of minimizing the boundary condition error. It can be seen that as the weight on the normal constraint on the surface increases, the error associated with the boundary condition decreases at the cost of introducing more error in the approximation of the grid. The effect of emphasizing some of the grid points more heavily than others is evident by comparing the two tables. In Table 3.2, with the leading edge points more heavily weighted, the errors associated with boundary condition are greater than in Table 3.1, in which all of the grid points are equally weighted. This difference is due to the increased emphasis on the accuracy of the approximation of the position of the leading edge grid points, resulting in surfaces which are closer to the leading edge of the initial surface.

Some of the results of the fittings are shown in Figures 3-7 to 3-10. By introducing weight on the normal constraint, a functionally designed blade (Figure 3-7) is created with greater camber than that of the original blade (Figure 3-6). Further weighting increases the camber more (Figure 3-8). Upon comparison with the initial blade, however, the leading edge appears to pull away from its initial position with increased weighting of the normal. To maintain the leading edge as close as possible to the leading edge of the initial blade, the position of the leading edge grid points are weighted more heavily. The results of weighting the leading edge can be seen in Figures 3-9 and 3-10, in which the displacements of the new leading edges from the initial leading edge are significantly reduced. The normals from the functionally designed blade in Figure 3-10 are illustrated in Figure 3-11, and are closer to the prescribed normals (Figure 3-4) than are the normals of the initial blade (Figure 3-3).

The second test of our fitting technique involves iterating the blade surfaces designed by the new method with flow analysis by PBD until it converges. An initial blade shape (Figure 3-12) is first analyzed by PBD producing a grid of points and corresponding velocity vectors on the surface. The new method is then used to fit this data with the prescribed normals weighted one and a half times the grid points and the leading edge and tip points weighted ten times the other grid points. The resulting surface is used as input to PBD which in turn produces another grid of points and velocity vectors. This process continues until no further significant improvement in satisfying Equation 2.4 is expected. The functional surfaces during the first, second,

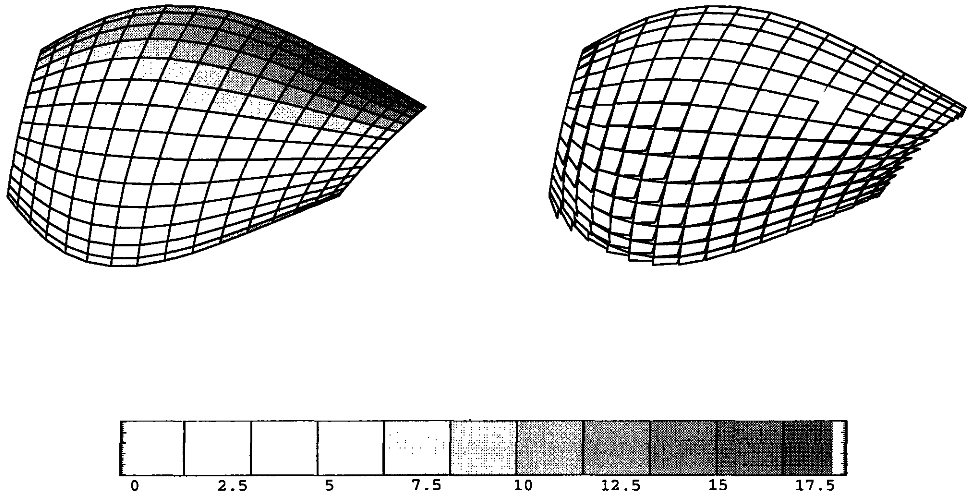


Figure 3-5: Reduction of angle between tangent plane of the blade surface and fluid velocity at the surface after four iterations of the design process. The shading shows the error in satisfying the kinematic boundary condition as measured in terms of this angle in degrees.

and third iterations of this process are shown in Figures 3-13 to 3-15. The errors associated with the blades in these figures are shown in Table 3.3. Notice that the difference between the fitting errors of successive iterations decreases. When this difference is small enough to be considered negligible, the design process is stopped as was the case between our third and fourth iteration.

As shown from the error data in Tables 3.1 and 3.2, a compromise must be made between approximating the grid points and the prescribed normals. As the error in the fitting of the normals decreases, the grid approximation error increases and vice versa, depending upon the weighting of the normal constraints. Greater weighting increases position error, moving the surface further away from where the prescribed normals (dependent upon the flow velocity at these positions) are considered relevant. Therefore, it is up to the designer to find the weights which produce the optimal results for the application.

	$\lvert\vec{S}(u_{i_p},v_{j_p}) - \vec{Q}_{i_pj_p}\rvert$		$\lvert(\vec{S}_u(u_{i_p},v_{j_p}) \times \vec{S}_v(u_{i_p},v_{j_p})) \cdot \vec{V}_{i_pj_p}\rvert$		$\angle(T_{\vec{S}(u_{i_p},v_{j_p})}, \vec{V}_{i_pj_p})$	
w	Average	Maximum	Average	Maximum	Average	Maximum
0.0	0.000189	0.000719	0.316499	1.401494	8.495245	23.902126
0.5	0.003810	0.011586	0.267714	1.304629	7.095615	20.240933
1.0	0.008635	0.023435	0.197120	1.233383	4.951475	19.045220
1.5	0.011889	0.029980	0.153630	1.199562	3.614010	18.553754
2.0	0.014109	0.033564	0.128116	1.162141	2.842993	17.954885
2.5	0.015822	0.035743	0.111388	1.128429	2.360101	17.523712
3.0	0.017270	0.037112	0.099107	1.084555	2.026850	16.868928
3.5	0.018556	0.039727	0.089406	1.081881	1.779790	16.916087
4.0	0.019724	0.044701	0.081488	1.080979	1.589116	16.901566
4.5	0.020796	0.049194	0.074968	1.074931	1.439891	16.804180
5.0	0.021785	0.053238	0.069656	1.062591	1.324160	16.605648

Table 3.1: Errors from functional surface fitting.

	$\lvert\vec{S}(u_{i_p},v_{j_p}) - \vec{Q}_{i_pj_p}\rvert$		$\lvert(\vec{S}_u(u_{i_p},v_{j_p}) \times \vec{S}_v(u_{i_p},v_{j_p})) \cdot \vec{V}_{i_pj_p}\rvert$		$\angle(T_{\vec{S}(u_{i_p},v_{j_p})}, \vec{V}_{i_pj_p})$	
w	Average	Maximum	Average	Maximum	Average	Maximum
0.0	0.000189	0.000740	0.316394	1.401205	8.492818	23.919587
0.5	0.003381	0.011600	0.293909	1.302441	7.809112	20.165175
1.0	0.007363	0.024075	0.249149	1.225216	6.413096	18.914309
1.5	0.010750	0.032323	0.210755	1.186805	5.205370	18.349376
2.0	0.013603	0.037664	0.183556	1.149414	4.339277	17.751676
2.5	0.015947	0.041263	0.164151	1.117463	3.726943	17.347980
3.0	0.017889	0.043775	0.149082	1.072330	3.271409	16.732444
3.5	0.019533	0.045588	0.136639	1.076268	2.915895	16.825698
4.0	0.020957	0.047599	0.126044	1.075508	2.629034	16.813471
4.5	0.022214	0.052566	0.116937	1.148366	2.393004	18.477771
5.0	0.023338	0.057075	0.109132	1.275384	2.197280	20.609227

Table 3.2: Errors from functional surface fitting with points on the leading edge weighted ten times as much as the other grid points.

	$\lvert\vec{S}(u_{i_p},v_{j_p}) - \vec{Q}_{i_pj_p}\rvert$		$(\vec{S}_u(u_{i_p},v_{j_p}) \times \vec{S}_v(u_{i_p},v_{j_p})) \cdot \vec{V}_{i_pj_p}$		$\angle(T_{\vec{S}(u_{i_p},v_{j_p})}, \vec{V}_{i_pj_p})$	
w	Average	Maximum	Average	Maximum	Average	Maximum
Input Blade			0.217072	0.453817	6.809438	18.122740
Iteration 1	0.008883	0.033422	0.063430	0.375692	1.654174	7.213728
Iteration 2	0.003073	0.009160	0.042570	0.344234	1.058455	6.588828
Iteration 3	0.001782	0.005836	0.037425	0.323409	0.916337	6.180788
Iteration 4	0.001408	0.005564	0.036323	0.302023	0.885490	5.767881

Table 3.3: Errors associated with the functional surface design of a propeller blade at different stages in the hydrodynamic analysis and surface fitting process.

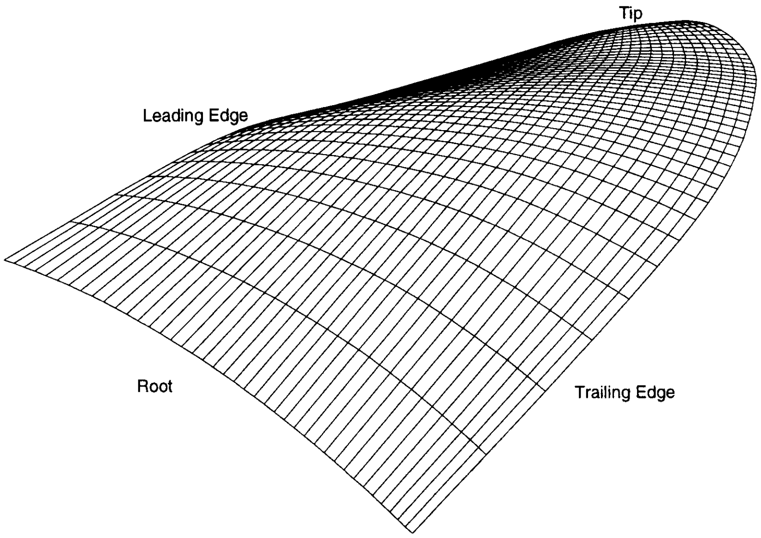


Figure 3-6: The initial blade shape used as input into the propeller analysis program.

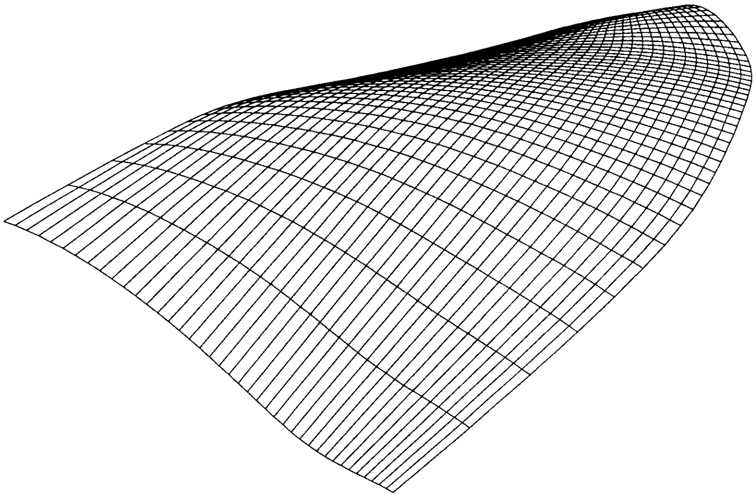


Figure 3-7: The requirement of the hydrodynamic boundary condition introduces more camber into the blade when the data is functionally fitted ($w = 1.5$ for the normal vectors).

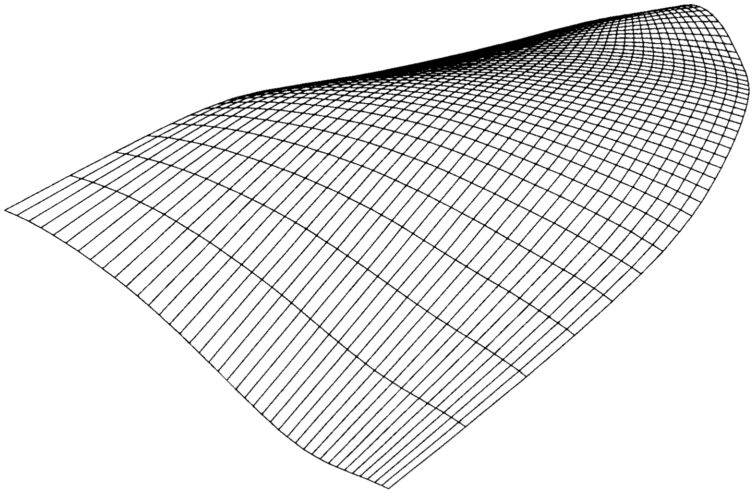


Figure 3-8: Increasing the weight of the prescribed normals further increases the camber of the blade ($w = 2.5$ for the normal vectors).

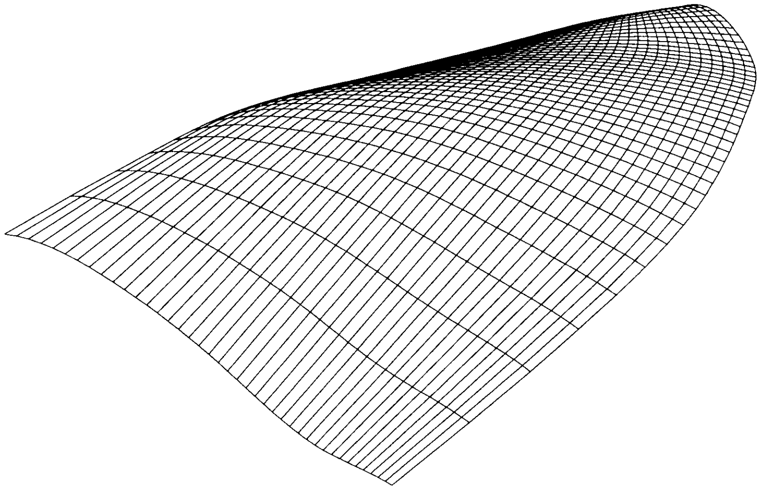


Figure 3-9: Applying increased weight (10 *times*) to the position data at the leading edge approximates the initial leading edge more closely ($w = 1.5$ for the normal vectors).

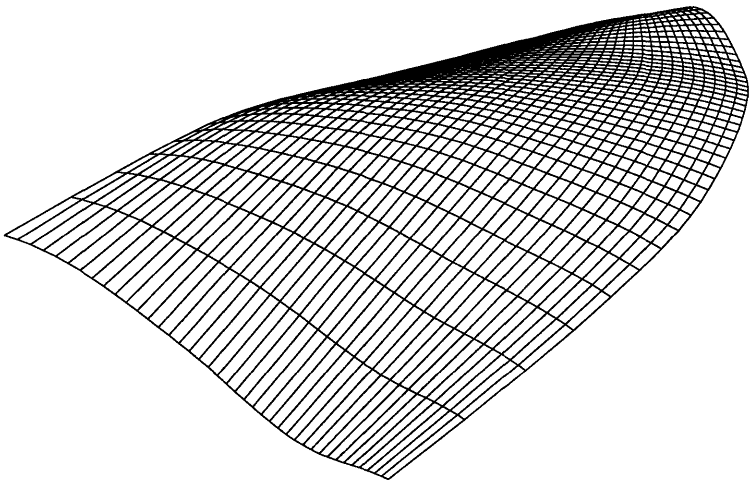


Figure 3-10: Increased prescribed normal weight ($w = 2.5$ for the normal vectors) with the leading edge positions weighted (10 *times*) more heavily than the other grid points.

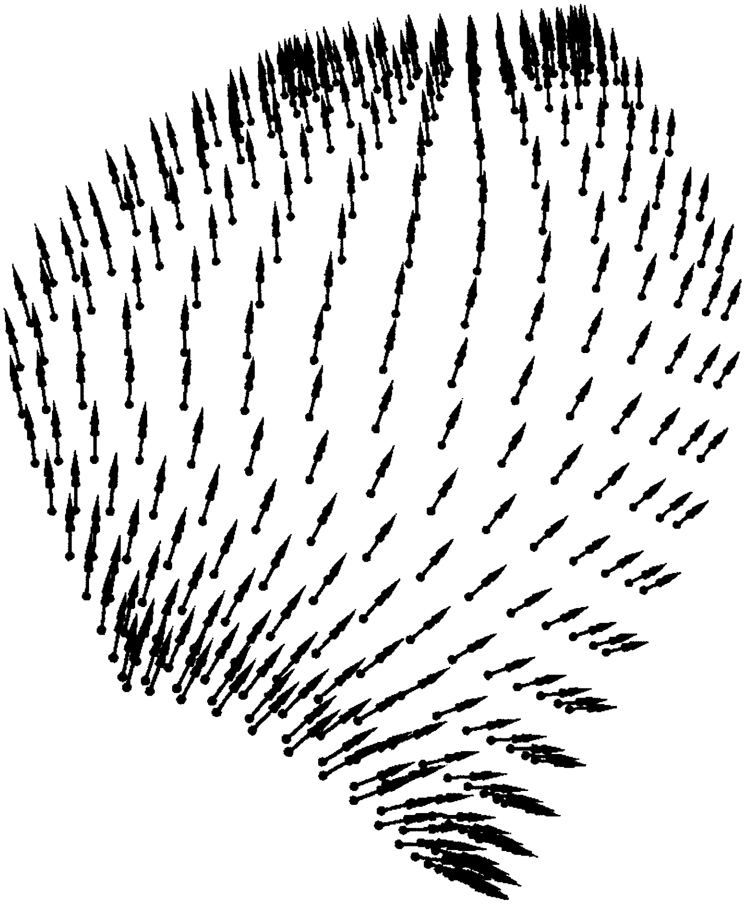


Figure 3-11: The field of normal vectors of the functional surface at the parametric locations (u_{i_p}, v_{j_p}). For this fitting, the normal vectors were weighed 2.5 *times* the effect of the interior position data and the leading edge position data were weighed 10 *times* more than position data at the interior of the blade.

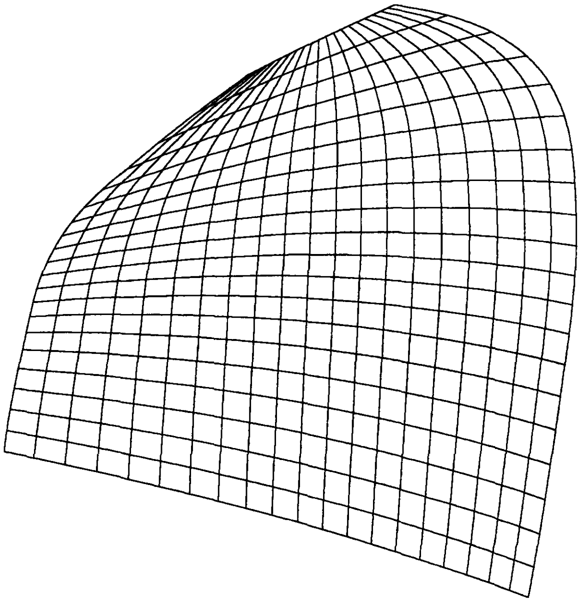


Figure 3-12: Initial blade shape to be analyzed in computer hydrodynamic simulation and improved through a series of shape modifying iterations.

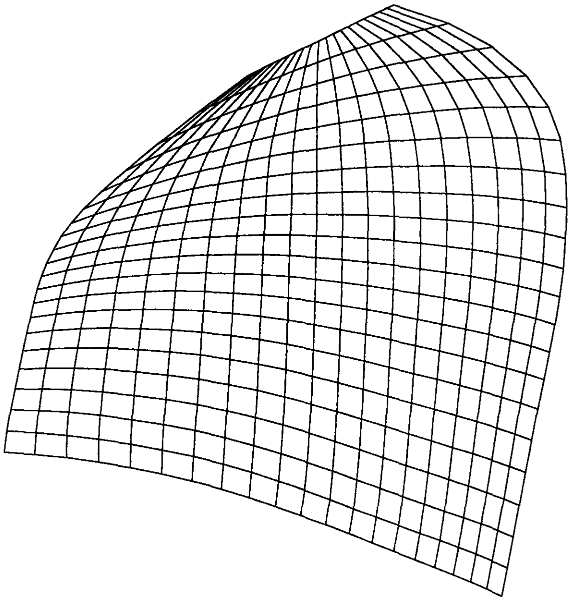


Figure 3-13: The functionally improved blade surface after one modification iteration, fitted to the output of the hydrodynamic analysis of the blade surface shown in Figure 3-12.

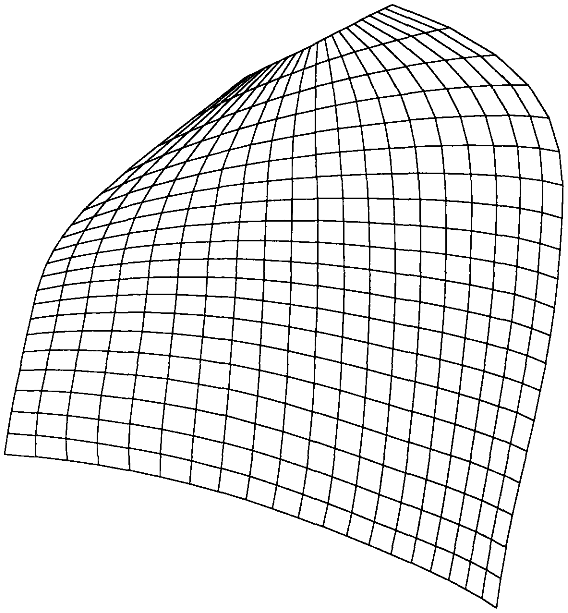


Figure 3-14: The functionally improved blade surface after two modification iterations, fitted to the output of the hydrodynamic analysis of the blade surface shown in Figure 3-13.

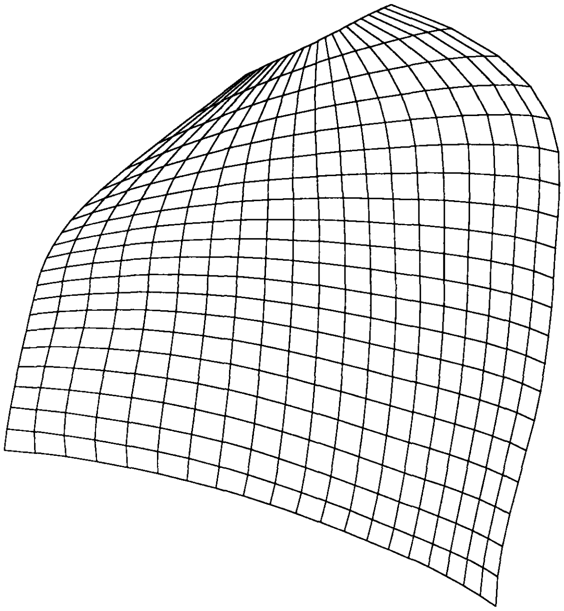


Figure 3-15: The functional blade surface after three hydrodynamic analysis and surface fitting iterations.

Chapter 4

Functional design with B-spline curves

4.1 Introduction

This chapter presents a complete design method for the functional design of mean camber lines of foil shapes described in terms of B-spline curves. This design method is an attempt to provide a robust method of functional design in which any lift distribution can be specified. The user inputs a desired (target) coefficient of lift of the mean camber line and the order of B-spline curve to be used for the geometric representation. For this method, the user need not provide a decent approximation of the final shape as an initial guess for the design process. The shape of the initial mean camber line is automatically set to represent a straight line of unit length satisfying all of the boundary conditions for uniform flow. By gradually modifying the shape to satisfy increasingly greater loads, a shape carrying the prescribed circulation distribution and total lift is found.

The shape modification process used in this design method differs from the process presented in the last chapter for designing propeller blades in two ways. First, additional constraints are applied to the positions of the node points (where the kinematic boundary condition is imposed), thereby reducing the number of degrees of freedom. This reduction of the number of degrees of freedom acts to prevent self-intersections and singularities from occurring during the design process. Second, a scaling constraint is introduced, preventing the curve from stretching or shrinking. With these two constraints, the number of degrees of freedom is reduced and an interpolation method can be used to find the new positions of the control vertices, as will be described below.

4.2 Representation of foil

Before describing the complete design process, the method of representing of the foil must be presented. As was stated before, the mean camber line of the foil is represented by a B-spline curve of arbitrary order with an associated knot vector and *control vertices.* In addition to the geometric description of the curve, additional properties need to be defined in order to facilitate functional design. These properties include the target coefficient of lift of the foil and circulation distribution from which the free stream velocity and total circulation about the foil are calculated. An additional property to be maintained internal to the fitting process is the loading, which is a coefficient indicating what fraction of the total desired circulation should be used when evaluating the flow. These hydrodynamic properties are then used to evaluate the flow velocity at the *node points* (i.e., the locations along the curve where constraints are applied during shape modification). These points are positioned parametrically midway between the nodes of the knot vector which are evaluated as the average of k consecutive knots. The circulation distribution along the foil is then divided up as discrete vortices which are positioned at these node points. From these lumped vortices and the free stream velocity, the velocity of the flow at the node points is evaluated. All of these properties are summarized in Table 4.1.

4.3 Design process

The design process developed for the functional design of mean camber lifting lines is illustrated in Figure 4-1. The first step of this design method involves initializing a

k	Order of curve.
m	Number of control vertices of curve.
$\vec{P}_i : 0 \leq i \leq m-1$	Position of control vertices.
$\mathbf{U}$	Knot vector of curve.
Γ	Target circulation about foil at 100% loading.
$\gamma(u)$	Circulation distribution over foil.
C_l	Target coefficient of lift at 100% loading.
l	Fraction of total loading at a given modification step.
$u_{i_p} : 0 \leq i_p \leq m-2$	Parametric locations of node points along the curve.
$\vec{Q}_{i_p} : 0 \leq i_p \leq m-2$	Position of node points.
V_∞	Free stream velocity.
$\vec{V}_{i_p} : 0 \leq i_p \leq m-2$	Velocity evaluated at the node points.
$\gamma_{i_p} : 0 \leq i_p \leq m-2$	Lumped vorticity associated with each node point.

Table 4.1: Properties associated with the functional design of a mean camber line represented in terms of a B-spline curve during the functional design process.

mean camber line and the inflow velocity. The user specifies a desired lift coefficient and the order of the curve. A vorticity distribution $\gamma(u)$ is also supplied; currently, this is predefined within the program although this can be modified to accept an arbitrary input file. The program then instantiates a B-spline foil of order k with $m \geq k$ control vertices, as specified by the user. An open, uniform knot vector $\mathbf{U}$ is automatically generated, as given by Equation 2.16. The control vertices are initially positioned such that the B-spline curve represents a line segment extending from $(0,0)$ to $(1,0)$. The free stream velocity V_∞ to be used during the design process is calculated by Equation 4.1, taking into acount the Kutta-Joukowski theorem (Equation 2.21), the total circulation Γ contained in the vorticity distribution $\gamma(u)$, and the definition of the coefficient of lift C_l.

$$\begin{aligned} L &= \frac{1}{2}\rho C_l V_\infty^2 = \rho V_\infty \Gamma \\ \Rightarrow \quad V_\infty &= \frac{2\int_0^{1.0} \gamma(u) du}{C_l} \end{aligned} \tag{4.1}$$

The last step in the initialization process is to set the loading to zero so that no circulation is associated with the current geometry. Hence, the foil starts as a straight line described in terms of a B-spline curve and satisfies all of the physical constraints

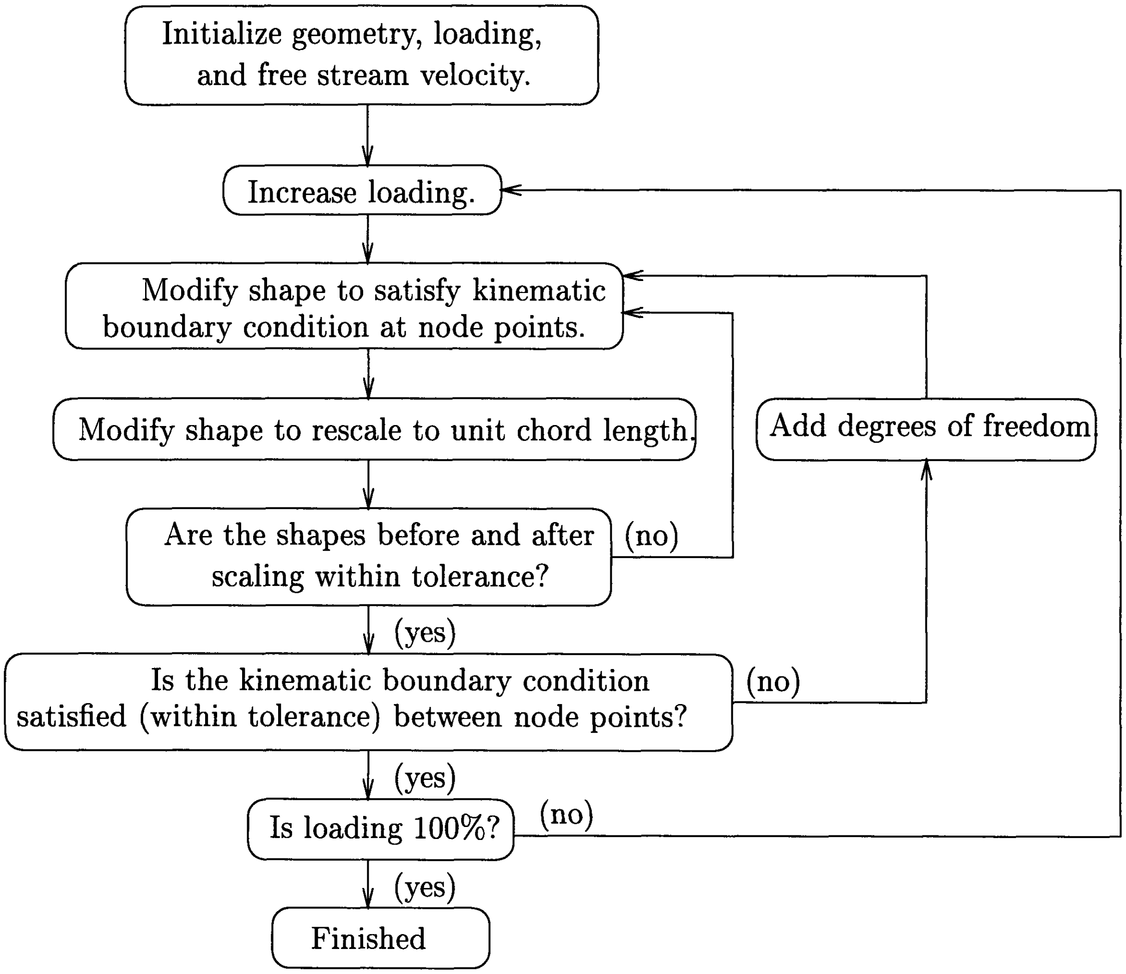

Initialize geometry, loading, and free stream velocity.
Increase loading.
Modify shape to satisfy kinematic boundary condition at node points.
Modify shape to rescale to unit chord length.
Add degrees of freedom
Are the shapes before and after scaling within tolerance?
(no)
(yes)
Is the kinematic boundary condition satisfied (within tolerance) between node points?
(no)
(yes)
Is loading 100%?
(no)
(yes)
Finished

Figure 4-1: Flow chart for the functional design of a mean camber line.

for a mean camber line generating zero lift, as shown in Figure 4-2.

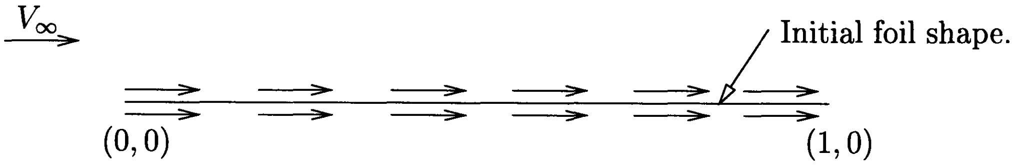


Figure 4-2: Initial mean camber line and flow in functional design process of the mean camber line of a foil.

The next phase of the design method progressively modifies the shape of the mean camber line by increasing the circulation until the desired coefficient of lift is reached. It consists of four steps: increasing the circulation, modifying the shape according to the kinematic boundary condition, rescaling, and adding additional degrees of freedom when needed. The steps of this loop will now be described in the order in which they are performed.

The first step in the loop is to increase the circulation over the foil. In increasing the circulation, the flow will no longer be parallel to the curve but will violate the kinematic boundary condition. The amount to increase the circulation must be chosen carefully. By increasing the circulation an amount which does not introduce an error (as measured by the angle between the velocity vector and the curve tangent at the node points) greater than a maximum allowable amount, the new curve shape after modification will be only slightly different than the preceding one. The new shape, however, will better satisfy the boundary condition for the increased loading. If the loading is allowed to increase too much in a single step, however, too great an angle between the velocity and the curve tangent can result in instabilities and self-intersections during the modification process, resulting in a physically unrealizable flow and foil shape. The loading is initially increased by some set amount (say 50% of the total loading) and then reduced until the error in satisfying the kinematic boundary condition for the current geometry is measured to be less than some permissible angle (say 5°). For each loading, the velocity at the node points is evaluated and the violation of the kinematic boundary condition is evaluated in terms of the angle formed between the curve's tangent and the velocity at these points. By only allowing the circulation to increase slightly, the current geometry will violate the kinematic boundary condition but the amount the shape will need to be adjusted can be controlled, as shown in Figure 4-3.

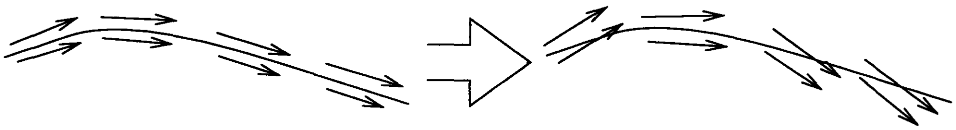


Figure 4-3: Given a mean camber line satisfying the physical constraints for a distribution of circulation, an increase in the circulation will introduce a violation of the kinematic boundary condition requiring the shape of the curve to be slightly modified.

Modification of the shape begins by allowing the curve to move in such a way as to better approximate the kinematic boundary condition. This is done by allowing the node points to be repositioned normal to the curve such that the tangent of the new curve at these points will be tangent to the velocity vectors measured at these node points' previous positions. By putting a limit on the maximum angle measuring the error in satisfying the boundary condition (in the previous step), the node points will only need to be adjusted slightly and the velocity vectors at their previous positions $\vec{Q}^*_{i_p}$ will be a good approximation to the velocity at their new positions $\vec{Q}_{i_p}$. In addition, since the tangent of the curve is affected by the relative position of the control vertices, the first control vertex can be kept fixed, thereby keeping the leading edge of the curve fixed throughout the modification process. This stage of the modification process can be stated as follows:

Given an initial B-spline curve representing a mean camber line of a foil with the control vertices $\{\vec{P}^*_i \ : \ 0 \leq i \leq m-1\}$ *and a set of velocity vectors* $\{\vec{V}_{i_p} : 0 \leq i_p \leq m-2\}$ *and sampled at node locations* $\{\vec{Q}^*_{i_p} = \vec{P}^*(u_{i_p}) \ : \ 0 \leq i_p \leq m-2\}$ *along the curve, reposition the control vertices* $\{\vec{P}_i \ : \ 1 \leq i \leq m-1\}$ *such that the curve at the parametric locations corresponding to the node points is now tangent to this set of velocity vectors. In addition, the new positions of the node points* $\{\vec{Q}_{i_p} = \vec{P}(u_{i_p}) \ : \ 0 \leq i_p \leq m-2\}$ *must lie along lines normal to the initial curve and passing through the previous node point positions* $\{\vec{Q}^*_{i_p} \ : \ 0 \leq i_p \leq m-2\}$.

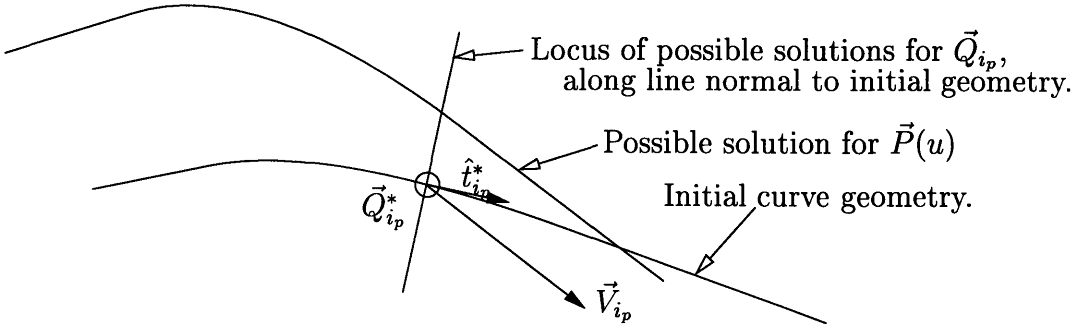

Locus of possible solutions for $\vec{Q}_{i_p}$, along line normal to initial geometry.
Possible solution for $\vec{P}(u)$
Initial curve geometry.
$\hat{t}^*_{i_p}$
$\vec{Q}^*_{i_p}$
$\vec{V}_{i_p}$

Figure 4-4: Example of the constraints imposed at one node point during the shape modification of a mean camber line.

In order to pose this problem mathematically, let us first define the vector $\delta\vec{P}_{i_p} = \vec{P}(u_{i_p}) - \vec{P}^*(u_{i_p})$ as the vector normal to the curve at node point $\vec{Q}^*_{i_p} = \vec{P}^*(u_{i_p})$. Also, let us assume we can prescribe a normal vector $\hat{n}_{i_p}$ which satisfies the physical constraint. The constraint on the position of the each node point and the physical constraint on the tangent $\vec{P}'(u_{i_p})$ of the new curve at each point can then be expressed as in equations 4.2 and 4.3, respectively.

$$\begin{aligned} \delta\vec{P}_{i_p} \cdot \hat{t}^*_{i_p} &= 0 \quad (i_p = 0, \cdots, m-2) && (4.2) \\ \vec{P}'(u_{i_p}) \cdot \hat{n}_{i_p} &= 0 \quad (i_p = 0, \cdots, m-2) && (4.3) \end{aligned}$$

where $\hat{t}^*_{i_p}$ is the tangent of the initial shape at u_{i_p}.

Substituting for the definition for $\delta\vec{P}_{i_p}$ and noticing that $\hat{n}_{i_p}$ must be perpendicular to the measured velocity vector $\vec{V}_{i_p}$, the equation system becomes

$$\begin{aligned} \left[\vec{P}(u_{i_p}) - \vec{Q}^*_{i_p}\right] \cdot \hat{t}^*_{i_p} &= 0 \quad (i_p = 0, \cdots, m-2) && (4.4) \\ \vec{P}'(u_{i_p}) \times \vec{V}_{i_p} &= 0 \quad (i_p = 0, \cdots, m-2). && (4.5) \end{aligned}$$

This is a linear equation system and can readily be solved using a standard matrix equation solver. To show this, Equations 4.4–4.5 are written in matrix form with the unknown control vertices on the left hand side:

$$\begin{aligned} \sum_{i=1}^{m-1} \left[\vec{P}_i B^k_i(u_{i_p}) \cdot \hat{t}^*_{i_p}\right] &= \left[\vec{Q}^*_{i_p} - \vec{P}_0 B^k_0(u_{i_p})\right] \cdot \hat{t}^*_{i_p} && (4.6) \\ & \qquad (i_p = 0, 1, \cdots, m-2) \\ \sum_{i=1}^{m-1} \left[\Big(P_{i,x} V_{i_p,y} - P_{i,y} V_{i_p,x}\Big) B'^k_i(u_{i_p})\right] &= \Big(- P_{0,x} V_{i_p,y} + P_{0,y} V_{i_p,x}\Big) B'^k_0(u_{i_p}) && (4.7) \\ & \qquad (i_p = 0, 1, \cdots, m-2) \end{aligned}$$

Each constraint yields $m-1$ equations resulting in a matrix system of $2m-2$ equations total. Since each control vertex has two degrees of freedom (x and y), $2m-2$ unknowns are present in the problem which, provided the coefficient matrix is well conditioned, can be solved for exactly. The solution represents a modified shape which better approximates the kinematic boundary condition for the given loading of the circulation distribution. The new shape will not satisfy the boundary condition exactly, however, since the shape has changed and the velocity along the curve will have changed accordingly. This is due to the fact that since the geometry of the problem has changed, the flow and position in space where the boundary condition must be satisfied have changed. It is for this reason that we maintain an upper limit on the increase in the loading of the circulation on the foil, thereby ensuring the velocity along the new shape to be reasonably close to the initial velocity.

Presumably, the shape of the mean camber line has been modified to better carry the prescribed loading. In the process of modifying the shape, however, the mean camber line may have stretched or shrunk and may no longer be of unit length. This is an undesirable degree of freedom in the design process as it may result in the foil shrinking and possibly degenerating to a point or stretching and possibly growing without convergence. To correct for this, the control vertices of the line must be rescaled. There can be several ways of doing this. The method employed here is to rescale the vertices such that all of the node points remain the same distance from the leading edge. For simplicity, the parametric value assigned to each node point is used as the distance it is to be positioned from the leading edge. Formally, the problem of rescaling can be stated as:

Given a B-spline curve $\vec{P}(u)$ *and node points* $\{\vec{Q}^*_{i_p} : 0 \leq i_p \leq m-2\}$ *positioned at parametric locations* $\{u_{i_p} : 0 \leq i_p \leq m-2\}$ *along the curve, reposition the control vertices* $\{\vec{P}_i : 1 \leq i \leq m-1\}$ *such that the new positions of node points* $\{\vec{Q}_{i_p} = \vec{P}(u_{i_p}) : 0 \leq i_p \leq m-2\}$ *are located at distances* $\{u_{i_p} : 0 \leq i_p \leq m-2\}$ *from the leading edge* $\vec{P}_0$.

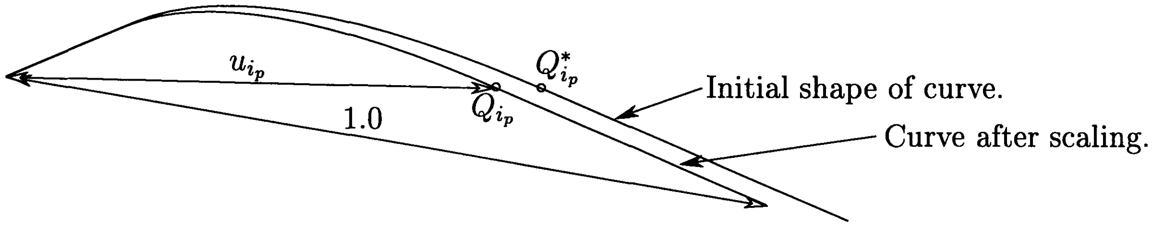


Figure 4-5: Example of repositioning the node points during functional design of a mean camber line to maintain a chord length of unity. Each node point $\vec{Q}_{i_p}$ is placed its respective distance of u_{i_p} from the leading edge.

This can be achieved by simply performing a linear interpolation of the desired new node point positions. To begin with, the new positions of the node points must be calculated at their respective distances from the leading edge, along on a line passing through the leading edge control vertex and the previous position of the node point. Let $\vec{Q}^*_{i_p}$ represent the position of the i_p^{th} node point found after the previous modification step involving the kinematic boundary condition, which may not be a distance u_{i_p} from the leading edge. Let $\vec{Q}_{i_p}$ be the new position of the node point which lies a distance of u_{i_p} from the leading edge along a line passing through the leading edge $\vec{P}_0$ and its previous position $\vec{Q}^*_{i_p}$. This rescaling process is illustrated in Figure 4-5. Again, the leading edge is kept fixed, fixing the curve in space. Mathematically, this rescaling step can be expressed as:

$$\vec{P}(u_{i_p}) = \vec{Q}_{i_p} \quad (i_p = 0, \cdots, m-2) \tag{4.8}$$

$$\vec{Q}_{i_p} = \frac{u_{i_p}}{|\vec{Q}^*_{i_p} - \vec{P}_0|}(\vec{Q}^*_{i_p} - \vec{P}_0) + \vec{P}_0 \quad (i_p = 0, \cdots, m-2) \tag{4.9}$$

The assumption being made here is that the distance each control vertex will have to be adjusted to rescale the mean camber line is relatively small relative to the chord length, thereby not tremendously altering the nature of the flow. Provided the control vertices were not moved exceedingly far during the previous modification (ensured by limiting the increase in the loading), this is a safe assumption.

Rescaling is simply followed by straightforward interpolation of the set of scaled nodes with a B-spline curve, and the solution can be found directly though a linear equation solver. The equations to be solved for the control vertices (except for the leading edge) are:

$$\sum_{i=1}^{m-1} \vec{P}_i B_i^k(u_{i_p}) = \frac{u_{i_p}}{|\vec{Q}^*_{i_p} - \vec{P}_0|}(\vec{Q}^*_{i_p} - \vec{P}_0) + \vec{P}_0\big(1 - B_0^k(u_{i_p})\big) \quad (i_p = 0, \cdots, m-2) \tag{4.10}$$

Again the number of unknowns $(2m-2)$ is balanced by the number of equations, and provided that the matrix is full-ranked, the Equation System 4.10 is solvable. Unlike the previous step, however, the problems for the x and y coordinates of $\vec{P}_i$ are uncoupled and only a smaller $(m-1) \times (m-1)$ system needs to be solved.

The rescaling step just described is purely geometric and does not involve the kinematic boundary condition. As a result, the rescaled shape may not satisfy the kinematic boundary condition as well as it might for the given number of control vertices and loading. To check to see how much the shape had to be adjusted to satisfy the scaling requirement, we compare the control polygon before and after the rescaling step. If any one control vertex had to be adjusted by more than an allowable maximum distance (say 1% of the chord length), then the velocity at the node points is recomputed and the boundary condition and scaling modification steps are performed again. Given that the loading has not changed, the mean camber line will converge to a shape which satisfies both the boundary condition and the scaling requirement to within some acceptable tolerance (as measured by the changes in the control polygon), at which point we move on to the next step.

So far, all constraints involving the shape design (the kinematic boundary condition and scaling) have been enforced at node points positioned at prescribed parametric values along the curve. Although the kinematic boundary condition may be satisfied at these points, there is no guarantee that it will be satisfied at other points along the curve. At this stage, a check is made at locations along the curve midway between consecutive node points. The angle between the velocity vector at these locations and the corresponding tangent of the curve is measured. If the error in satisfying the physical constraint (measured by this angle) at any one of these positions is greater than some allowable tolerance (say 5^o), more degrees of freedom are added and a new solution found which better satisfies the current circulation distribution over the foil. These additional degrees of freedom are introduced by inserting a new knot between each pair of consecutive, non-repeating knots in the knot vector of the curve. With each new knot, a new control vertex is added providing two additional degrees of freedom. New node points, accordingly, are also created providing the additional points on the curve where constraints must be satisfied. After all of the new knots are added, the foil shape is modified as outlined above for the same loading, yielding a shape which will better approximate the kinematic boundary condition over a larger portion of the foil. Once a solution is found which satisfies the boundary condition acceptable at both the node points and the locations midway between them, the whole process is repeated until the loading reaches 100%, generating the target coefficient of lift.

Each time through the loop outlined above, a foil mean camber line is found which will carry a given circulation distribution within a prescribed tolerance. Progressively the loading is increased each time through the loop, until the load on the foil is 100%, and the foil is generating the desired coefficient of lift. At this point, the design process is finished and a converged solution is found. This solution represents a mean camber line which will generate the specified coefficient of lift and carry the prescribed loading in terms of a B-spline curve.

4.4 Examples of the functional design of mean camber lines

In this section, some examples are presented, demonstrating the functional design of mean camber lines.

4.4.1 Effect of tolerance threshold on the functional design of mean camber lines.

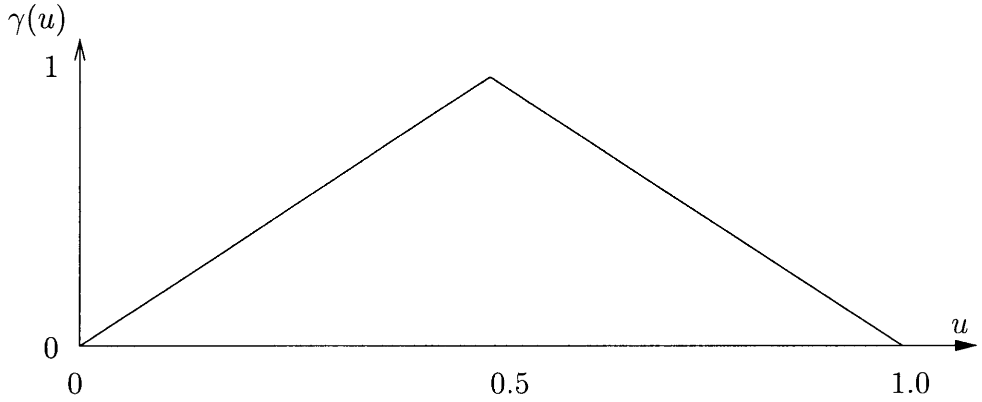


Figure 4-6: Saw tooth circulation distribution over length of mean camber line.

The first two mean camber lines to be considered are both designed to produce the same coefficient of lift with the same circulation distribution, as given in Figure 4-

6. A different tolerance threshold for the error in satisfying the kinematic boundary condition, however, was used resulting in slightly different results, illustrating the effect of adding degrees of freedom to introduce more shape flexibility to a design. In both cases, the design process started with a cubic (4^{th} order) B-spline curve with $m = 4$ control vertices and the target coefficient of lift was $C_l = 1.5$. As previously described, the curve shape was initialized as a straight line segment of unit length described in terms of a B-spline curve with the desired order and number of control vertices, as listed in Table A.1. Maximum allowable tolerance ϵ defined as the angle between velocity vectors at the node points and the tangent of the curve is the only difference between the two examples. The first curve, given in Figure 4-7 and Table A.2, is functionally designed with a tolerance of $\epsilon = 5^o$. In the design process of this curve, three knots had to be inserted to provide the additional degrees of freedom (control vertices) to satisfy the tolerance specified. The velocity vectors at the 6 node points are also shown in Figure 4-7. For the same target coefficient of lift and circulation distribution, a slightly more refined shape results if the tolerance is decreased to 3^o as shown in Figure 4-8 and Table A.3. To satisfy this decrease in tolerance, several more knots had to be inserted thereby increasing the curve's degrees of freedom to $m = 19$. With the added degrees of freedom, additional node points are needed as shown in Figure 4-8 by the increased number velocity vectors along the curve. The difference in the two designs can be seen upon comparison of Figure 4-7 and Figure 4-8. The design satisfying a stricter tolerance contains a tighter radius of curvature midway along the curve.

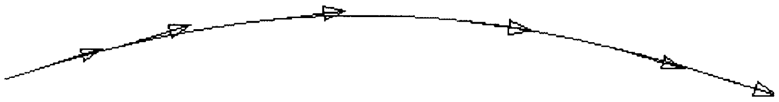


Figure 4-7: Mean camber line designed to generate loading distribution shown in Figure 4-6 with $C_l = 1.5$ and $\epsilon = 5^o$. Velocity at the node points is also illustrated.

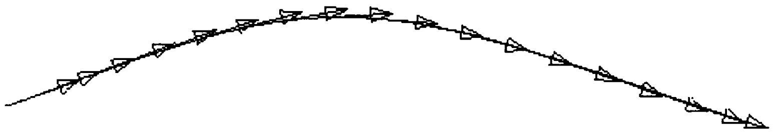


Figure 4-8: Mean camber line designed to generate loading distribution shown in Figure 4-6 with $C_l = 1.5$ and $\epsilon = 3^o$. Velocity at the node points is also illustrated.

4.4.2 Effect of coefficient of lift on the shape of a mean camber line.

To illustrate the effect the specification of the coefficient of lift has on the functional design of a mean camber line, the same circulation distribution given in Figure 4-6 is used to generate a foil shape with a coefficient of lift of $C_l = 2.0$. Again, the design process starts with the same straight line represented as a cubic B-spline curve given in Table A.1. The final curve shape and the velocity vectors at the node points is given in Figure 4-9 and Table A.4. To generate the higher coefficient of lift for the line given in Figure 4-9, the functional design process introduced more camber than for a foil with a coefficient of lift of only $C_l = 1.5$ (Figure 4-7). In addition, to achieve this greater camber, degrees of freedom had to be added to keep the design within tolerance. Whereas the final shape of the curve with $C_l = 1.5$ has only $m = 7$ control vertices to satisfy the functional requirements, the curve producing the greater coefficient of lift $C_l = 2.0$ needs $m = 11$ control vertices.

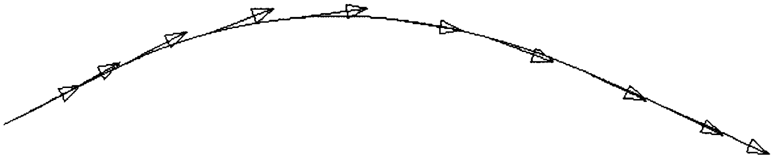


Figure 4-9: Mean camber line designed to generate loading distribution shown in Figure 4-6 with $C_l = 2.0$ and $\epsilon = 5^o$.

From these examples, the satisfaction of the kinematic boundary condition can be seen. The velocity at the node points is tangent (within the specified tolerance) to the curves, thereby satisfying the physical constraint with which the curve was designed. By either lowering the tolerance or increasing the coefficient of lift, degrees of freedom will need to be added to provide enough shape flexibility to satisfy the physical constraint.

4.4.3 Effect of curve order on the functional design of a mean camber line of a hydrofoil.

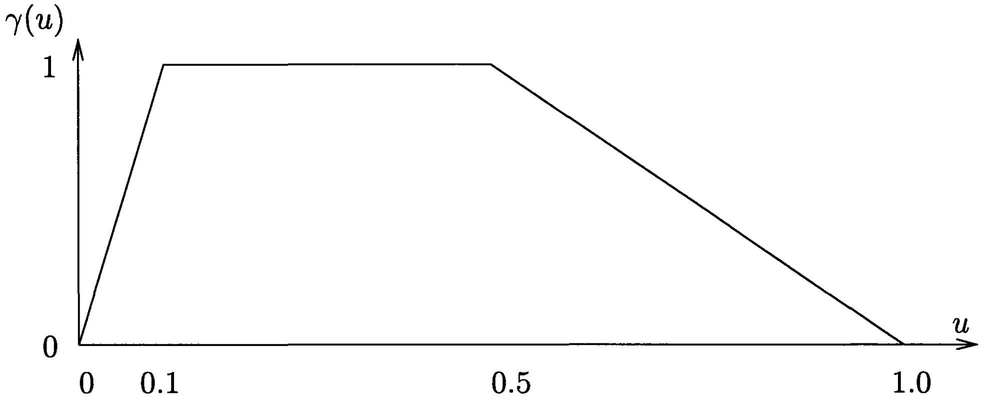


Figure 4-10: Circulation distribution producing nearly constant loading over the first half of the foil, linearly varying to zero over second half.

The next examples illustrate the effect of the curve's order on the shape found through the functional design process. All the curves produce a coefficient of lift of $C_l = 1.5$ with the circulation distribution given in Figure 4-10. As before, the process starts from a straight B-spline curve of the specified order with $m = 6$ control vertices. The shapes of the mean camber lines are given in Figures 4-11 to 4-14 and the corresponding data are given in Tables A.5 to A.14. The data for the initial shapes for each design are given in Tables A.5, A.9, A.11, and A.13. The final design data are listed in Tables A.6 to A.8, A.10, A.12 and A.14. Upon comparison of the four curves, one can readily observe that the shapes are very similarly despite the difference in the order of the curves. To satisfy the kinematic boundary condition, however, more degrees of freedom had to be added to the lower order curves than for the higher order curves for the given tolerance ($\epsilon = 5^o$). In fact, for the 2^{nd} order curve (which is piecewise linear) given in Tables A.6–A.8 a total of $m = 161$ control vertices were needed! For the 3^{rd} order, quadratic curve given in A.10, only $m = 10$ control vertices were used. For the higher order curves $k = 4$ and $k = 5$ given in A.12 and A.14, respectively, no additional degrees of freedom beyond the six control vertices initially provided were needed to satisfy the boundary condition. To capture the nature of the flow in the design, it would appear that a cubic curve provides sufficient shape flexibility. During the physical constraint check midway between node points, the lower degree curves did not naturally follow the flow about the foil and thus required the insertion of more degrees of freedom–more control vertices. The higher degree curves followed the nature of the flow between node points, requiring fewer degrees of freedom to satisfy the physical constraint.

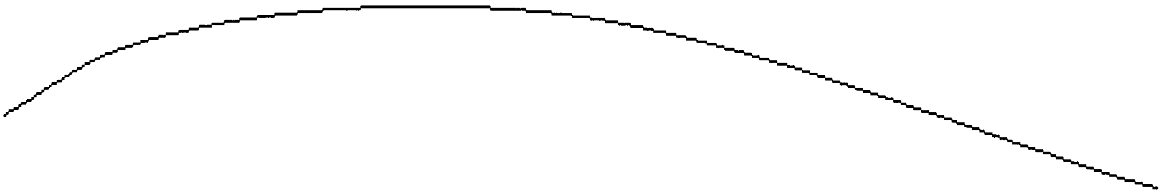


Figure 4-11: Mean camber line designed to generate loading distribution shown in Figure 4-10 with $C_l = 1.5$ using a B-spline curve of order $k = 2$.

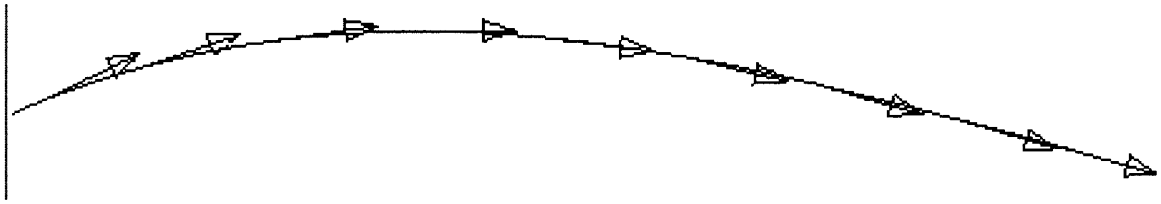


Figure 4-12: Mean camber line designed to generate loading distribution shown in Figure 4-10 with $C_l = 1.5$ using a B-spline curve of order $k = 3$. Velocity vectors are shown at the node points.

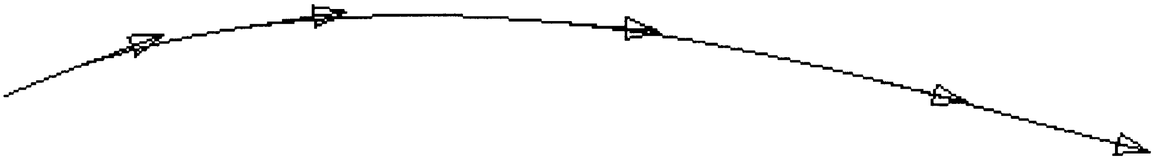


Figure 4-13: Mean camber line designed to generate loading distribution shown in Figure 4-10 with $C_l = 1.5$ using a B-spline curve of order $k = 4$. Velocity vectors are shown at the node points.

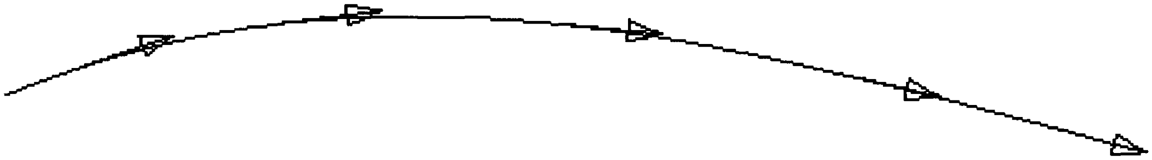


Figure 4-14: Mean camber line designed to generate loading distribution shown in Figure 4-10 with $C_l = 1.5$ using a B-spline curve of order $k = 5$. Velocity vectors are shown at the node points.

Chapter 5

Related applications

Fitting a set of normal vectors as well as a set of data points is not limited to the functional design of lifting surfaces. Two other possible applications are in the areas of data fitting and offsets. In data fitting, constraints on the fitting curve's normals can produce a curve more representative of the trends in the data. For offset curves and surfaces, normals from the progenitor can be used to produce a more accurate approximation to an offset or parallel curve. This chapter formulates these ideas and provides some examples.

5.1 Shape preserving interpolation of 2D data with B-splines

Interpolation of data with B-splines is a fairly straightforward process. Parameter values are assigned to a set of points and the shape of a B-spline is found such that the spline passes through these points at these parameter values:

$$\vec{P}(u_{i_p}) = \vec{Q}_{i_p} \qquad (i_p = 0, \cdots, m-1). \tag{5.1}$$

Equation 5.1 can be formulated as a linear equation system and the positions of the control vertices of the B-spline can be computed directly. There is no guarantee, however, that the interpolating spline will preserve the same intrinsic "shape" of the corresponding data set. By shape preservation, we mean that the variation in the shape of the curve follows the implied variations in the data. Where the variation in the data is monotonously increasing/decreasing, the curve should be also. Traditional methods of interpolation do not guarantee this. Consider, for example, the data set given in Figure 5-1. Two possible interpolating curves which do not preserve the shape of the data are shown in Figures 5-2 and 5-3. In the former case, the curve overshoots an intrinsic corner and in the second, the curve oscillates where the data implies a smoother form. The most desirable curve fit for both design applications and fitting of scientific data, is the one which preserves the implied shape of the data [16, 17, 39], without unintended oscillations or overshoots.

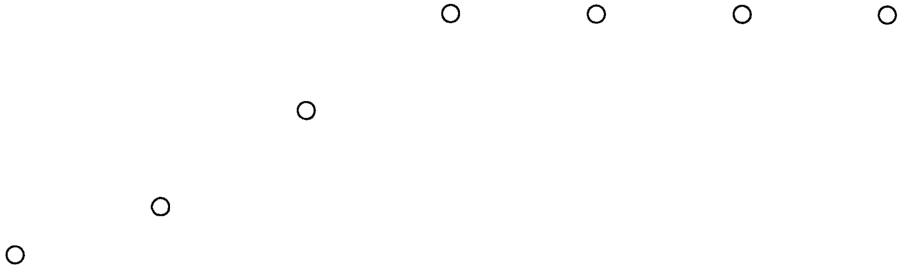


Figure 5-1: Data set to be interpolated: $\vec{Q}_i$.

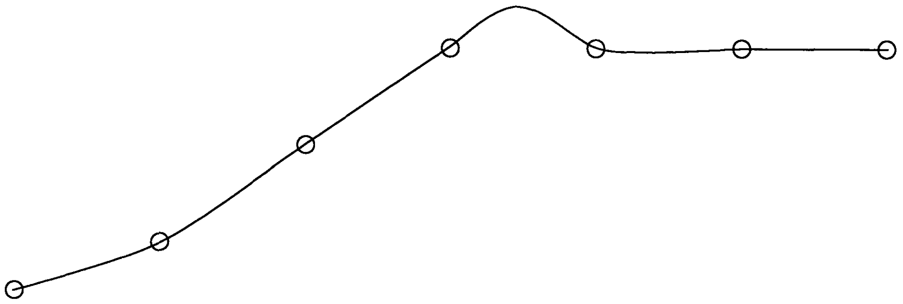


Figure 5-2: Interpolating curve overshooting corner implied by data set.

Previous methods of shape preserving interpolation involve a procedural approach. Desired curve tangents are computed at the data points and piecewise, Hermite curves are fit to the data maintaining G^1 continuity between segments [17, 39]. Another approach is to use a global method in which a tensioning parameter to each of the data points is first assigned. Then, a variational technique is used in which the tension

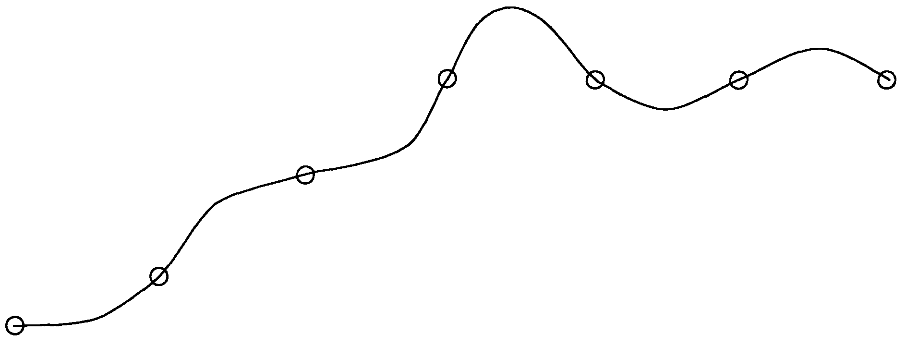


Figure 5-3: Interpolating curve oscillating more than data set.

of the curve at the data points is selected to best preserve the shape implied by the data.

The condition of orthogonality developed in this thesis to satisfy the kinematic boundary condition can be adapted to shape preserving interpolation. This is accomplished by assigning a desired normal to the curve at each of the data points and then interpolating both the data points and the specified normals:

$$\vec{P}(u_{i_p}) = \vec{Q}_{i_p} \qquad (i_p = 0, \cdots, m-1) \tag{5.2}$$

$$\vec{P}'(u_{i_p}) \cdot \hat{n}_{i_p} = 0 \qquad (i_p = 0, \cdots, m-1) \tag{5.3}$$

The solution is a curve $\vec{P}(t)$ which interpolates both the set of data and normals. Equation System 5.2–5.3 is a linear equation system and, provided that the normals are well chosen, the solution can be directly found through traditional matrix solvers. In this method, an arbitrary degree of continuity can be maintained. In fact, any parametric curve representation can be used (polynomial, spline, etc.) for which the tangent can be easily computed. B-spline curve representation was used to demonstrate the concept since this allows the user to specify an arbitrary degree of continuity but maintains local support.

The selection of the prescribed normals and knot vector is still an outstanding problem. Two possible approaches for determining a set of normals have been considered. The first is to fit a quadratic, Bézier curve to every three consecutive data points and use the normals of the curve at the middle data points as the prescribed normal for the curve interpolating all of the data. In this way, the prescribed normals are computed based upon the local variations in the curve thereby insuring shape preservation. The second method involves taking the normal at each interior point as the normal to a circle interpolating every three consecutive data points. For both methods, the normals at the ends of the curve are taken as normals to straight lines interpolating the end data points and their neighbors. Examples of computing the desired normal vectors by each method are shown in Figures 5-4 and 5-5.

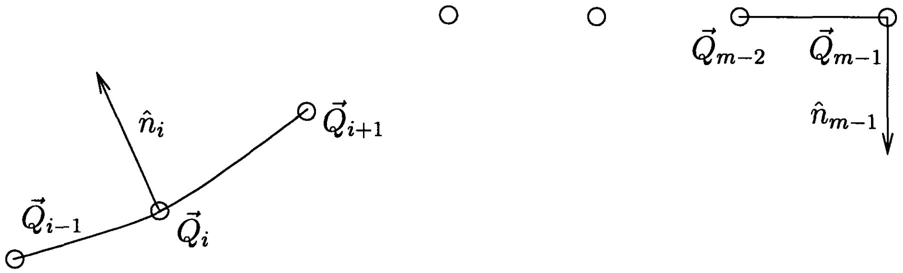


Figure 5-4: Computation of desired normals for based upon the normal to a quadratic Bézier curve interpolating the data.

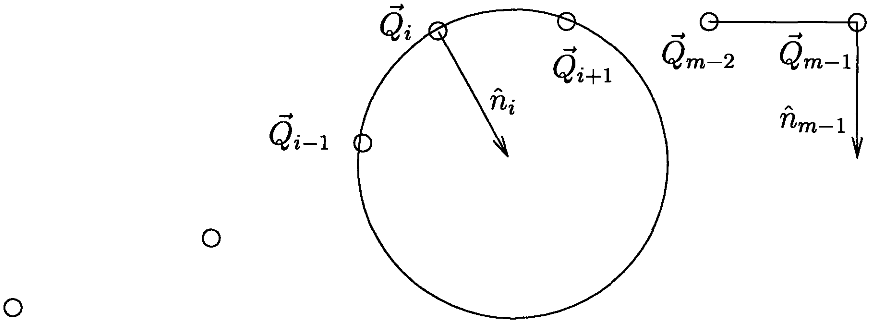


Figure 5-5: Computation of desired normals for based upon interpolating circles.

With the normals specified, a curve can be found interpolating both the data set and satisfying the orthogonality condition imposed by the normal vectors at the data points. Assuming that the prescribed normals reflect the intrinsic trends in the data set, the resulting curve should tend to preserve the form of the data. The task of computing the most appropriate knot vector is still left unresolved. In the implementation of this data fitting approach, the knot vector is determined interactively by the user. With further research, perhaps an automatic method to compute the optimal knot vector based upon the data and normal sets can be determined.

An example of fitting data with this approach is shown in Figure 5-6. The normals are determined at the data points according to the Bézier method illustrated in Figure 5-4. The tighter curve (the one with fewer oscillations) was produced through the interpolation of the data points and the normals. The second curve (with unwanted oscillations and overshoots) was generated through standard interpolation of the position data using a standard fitting routine supplied in the Praxiteles library [1, 21]. Even though the constrained interpolation presented in Figure 5-6 produces a tighter fit (more closely resembling the intrinsic form of the data than standard interpolation), user interaction was required to adjust the position of the knots, thus creating the more desirable curve.

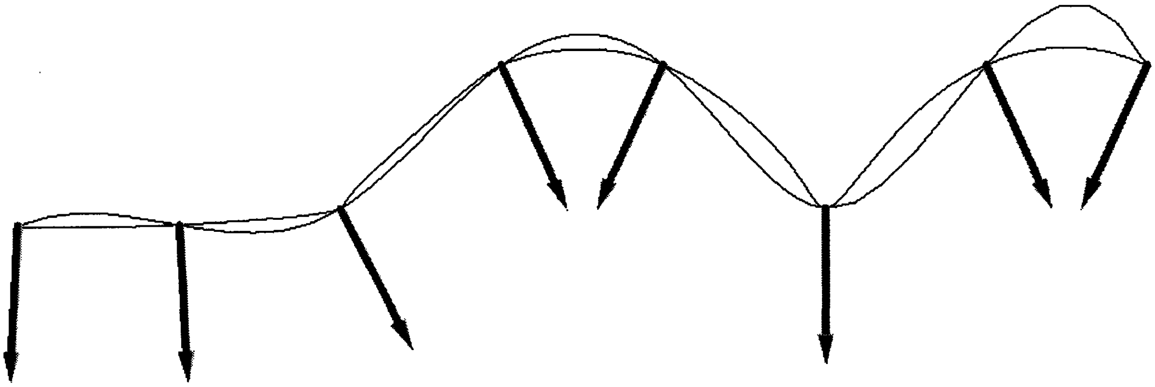


Figure 5-6: Interpolation of a data set by traditional interpolation and constrained interpolation with the prescribed normals shown. The constrained curve provides a "tighter" fit to the data.

5.2 Approximation of offsets of planar curves

Another possible application for the interpolation of a set of data points and a set of normals is in the approximation of offset or parallel curves and surfaces. An offset curve or surface is defined in terms of a progenitor curve or surface and an offset distance. For an offset curve, each point along the curve is determined by moving predetermined distance from the progenitor curve along normal lines to the progenitor [32]. Similarly for the offset surface. Some of the engineering applications for offset curves and surfaces are shown in Figure 5-7 including planning tool paths for NC machining [9, 15], describing tolerance regions [36, 30], determining access space for motions of robots [24], representation of curved shells in solid modeling [33], and feature extraction from the Medial Axis Medial (MAT) [31, 43].

Offset curves and surfaces of B-spline progenitors are functionally more complex than B-spline functions. If the CAD/CAM system processing and interrogation is based on the properties of B-splines, offsets need to be approximated with B-splines to permit their further processing. Current techniques for approximating offset curves involve sampling the progenitor curve over its length, interpolating points offset a prescribed distance normal to the progenitor at these sampled points, and then checking if the offset curve is within the acceptable tolerance of the exact offset curve. The density of the sampling is increased and the process repeated until a satisfactory offset is found.

Since the offset curve should have the same unit tangent at each of the sampling points as the progenitor, the interpolation of a set of normals could provide a more accurate approximation of the exact offset curve. This can be accomplished by imposing the condition of orthogonality developed for functional shape design on the offset curve's tangent with respect to the sampled normal of the progenitor curve. Therefore, instead of interpolating simply a set of points, the offset curve will also satisfy a set of perpendicularity conditions as well, ensuring that both curves will be parallel at these points. This process of constructing an offset curve through this technique can be described more precisely as follows:

Given a progenitor curve $\vec{P}^*(u)$*, sample the position and normal of the curve uniformly to create a set of data points* $\{\vec{P}^*(u_{i_p}) = \vec{Q}^*_{i_p} : 0 \leq i_p \leq m_p - 1\}$ *and normal vectors* $\{\hat{n}_{i_p} : 0 \leq i_p \leq m_p - 1\}$*. Next, offset each data point by a constant distance d along its respective normal from its initial position* $\{\vec{Q}_{i_p} = \vec{Q}^*_{i_p} + d\hat{n}_{i_p} : 0 \leq i_p \leq m_p - 1\}$*. The approximate offset curve* $\vec{P}(u)$ *is the interpolation of the data points as well as the sampled normal vectors* $\{\vec{P}(u_{i_p}) = \vec{Q}_{i_p},\ \vec{P}'(u_{i_p}) \cdot \hat{n}_{i_p} = 0 : 0 \leq i_p \leq m_p - 1\}$

Mathematically, this discrete formulation of computing approximate offset curves

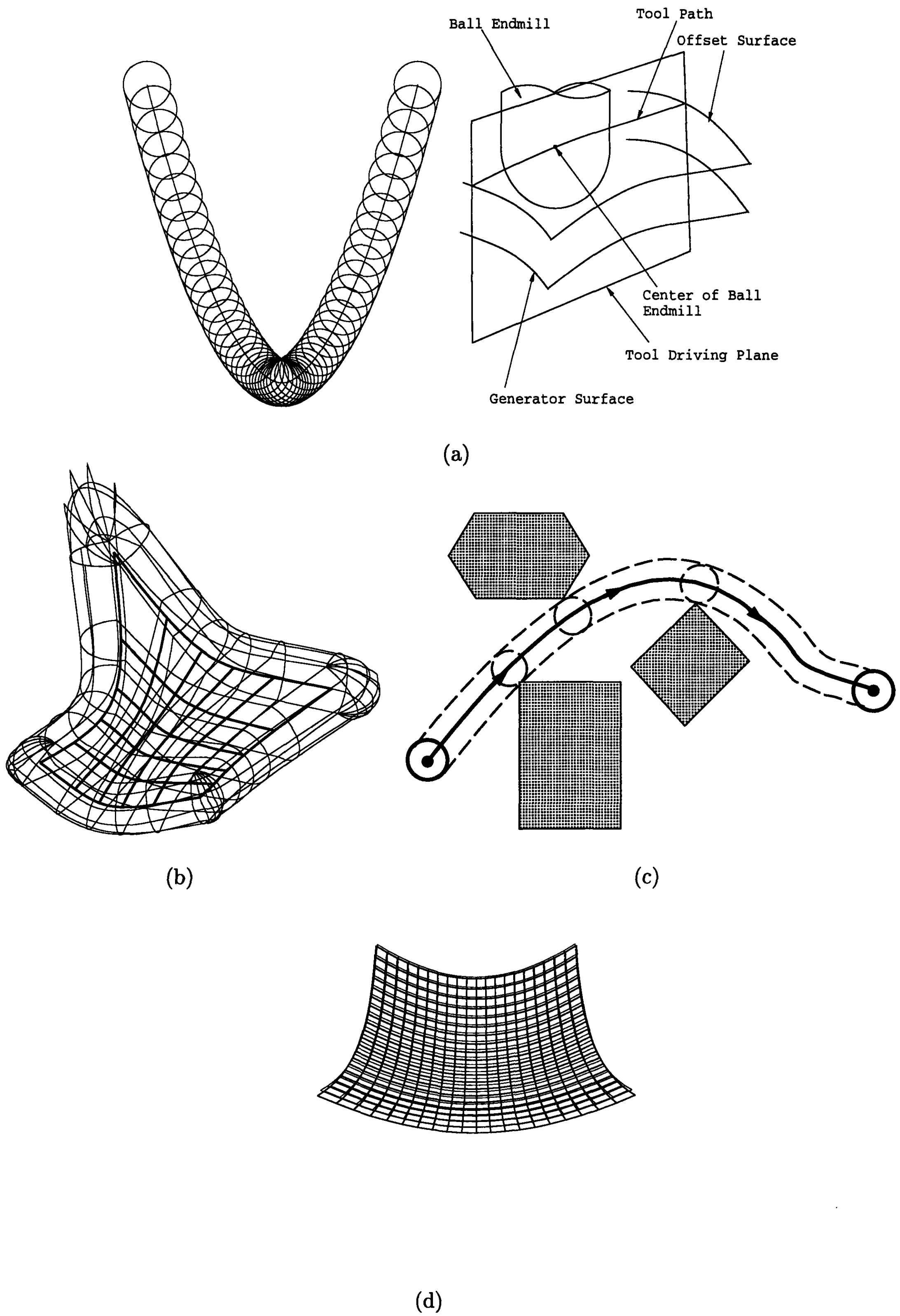


Figure 5-7: (a) NC machining; (b) Tolerance regions; (c) Access space representation in robotics; (d) Plate/shell representation. (Adapted from Patrikalakis and Maekawa [32])

$\vec{P}(u)$ becomes:

$$\vec{P}(u_{i_p}) = \vec{P}^*(u_{i_p}) + d\frac{\vec{P^*}'(u_{i_p}) \times \hat{k}}{|\vec{P^*}'(u_{i_p})|} \quad (i_p = 0, 1, \cdots, m_p - 1) \quad (5.4)$$

$$\vec{P}'(u_{i_p}) \cdot \frac{\vec{P}'(u_{i_p}) \times \hat{k}}{|\vec{P}'(u_{i_p})|} = 0 \quad (i_p = 0, 1, \cdots, m_p - 1) \quad (5.5)$$

where $\hat{k}$ is the unit vector normal to the plane of the curve. The progenitor and offset curves are represented by $\vec{P}^*(u)$ and $\vec{P}(u)$, respectively. The parametric values at which interpolation is enforced are $\{u_{i_p} : 0 \leq i_p \leq m_p - 1\}$. As in the functional design of hydrofoil mean camber lines, additional control vertices are needed in order for the numbers of equations and unknowns to balance. Equation System 5.4 provides 2 equations (for the x and y dimensions) and Equation System 5.5 only provides 1 equation for each sample. Therefore, the complete system is balanced if the number of unknown control vertex coordinates $2m$ equals the number of constraining equations $3m_p$, where m is the number of control vertices and m_p is the number of sampled points along the progenitor. Due to this fact, the number of samples m_p of the progenitor curve is required to be an even number, i.e. $m_p = 2m'_p$ and therefore $m = 3m'_p$.

Some examples of approximate offset curves computed by this method are shown in Figure 5-8. No tolerance threshold is enforced–the offset curves are the results of a single interpolation of a set a data points and the corresponding normals from the progenitor curves. For simplicity, the knot vector for the offset curve is chosen to be a uniform knot vector. Despite the fact that the number of equations equals the number of unknowns, the system of equations may still be ill-conditioned or even singular. This may be due to the position of the offset data points, directions of the normal vectors, or choice of knot vector.

Although this method only approximates the exact offset geometry, the interpolation of the normal vector as well as offset data points provides more information about the shape of the offset curve for a given number of sampling points. Rather than needing to more densely sample a curve to accurately represent the tangent of

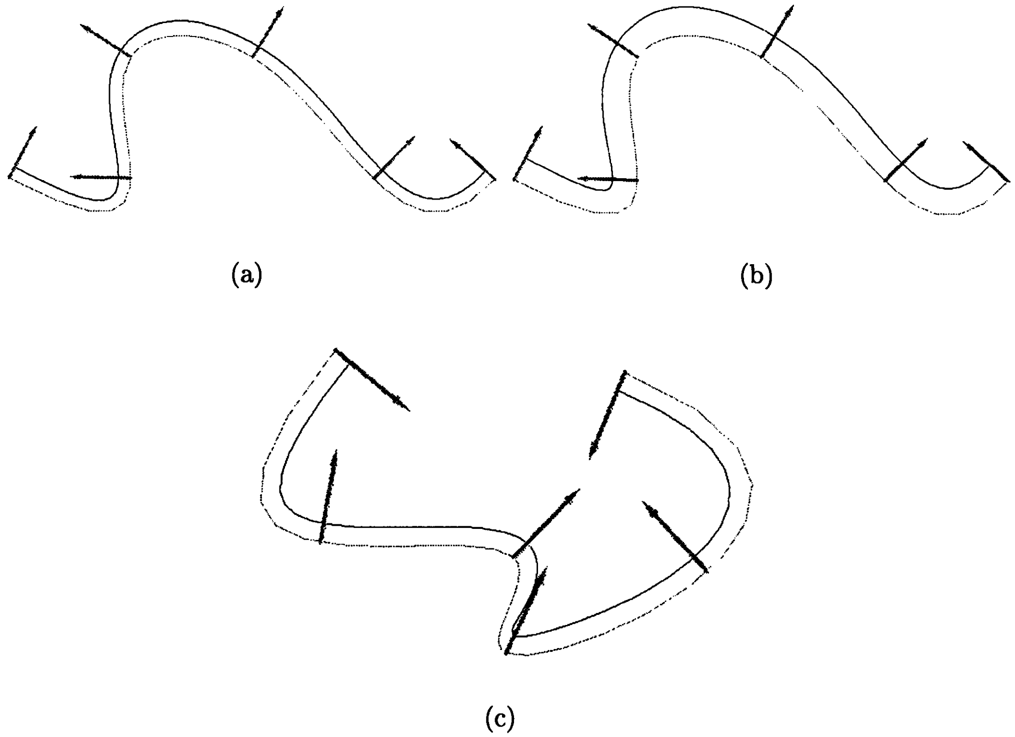


Figure 5-8: Samples of approximate offset curves generated by interpolating a set of data points and normal vectors. With relatively fewer samplings of the progenitor curve, an acceptable offset curve approximation can be found with normal constrained fitting.

the offset as in traditional methods, this approach allows the exact tangent to be maintained at a discrete set of points. Although this method is presented in terms of planar B-spline curves, it is easily extensible to space curves, B-spline surfaces, and other more general parametric representations of curves and surfaces.

Chapter 6

Conclusions and recommendations

6.1 Conclusions

This thesis presented methods to functionally design hydrodynamic lifting surfaces. Whereas geometric design methods may only involve the interpolation of data sets and fairness criteria, functional design incorporates the physical constraints determining the performance of a shape directly into the design process. For the lifting curves and surfaces considered here, the physical constraint is the kinematic boundary condition which is considered in conjunction with a potential flow method for describing the ideal flow about a lifting surface. To create a new hydrofoil shape, the designer prescribes a desired pressure (or circulation) distribution about an initial shape, dealing directly with the physical nature of the flow (lift) rather than the foil's geometry. A new shape is then functionally designed to better carry this prescribed loading by finding a new blade or foil shape which better satisfies the kinematic boundary condition at a discrete set of points along its shape.

The first method presented uses an approximation approach for the modification of a mean camber surface. The hydrodynamics code developed by Kerwin *et al.* [22] is used to evaluate the flow along the mean camber surfaces of a propeller blade. The blade shape probably will not carry the prescribed pressure distribution exactly resulting in some degree of violation of the kinematic boundary condition. From the velocity vectors at a discrete grid of points of the blade, a set of normals are computed which would better satisfy the physical constraint by a method developed in this thesis. Next, a new surface is fit to the set of data points and prescribed normals, resulting in a modified blade shape better carrying the specified loading. The novelty of fitting a prescribed set of normal vectors satisfying the kinematic boundary condition is that the problem is transformed without approximation to a linear problem. Therefore, standard linear equation system solvers can be used and solutions found quickly, robustly, and efficiently. In this approach, however, the violation in the kinematic boundary condition must not be too excessive so that only a slight alteration of the blade shape is needed (the prescribed normals deviate only slightly from the normals sampled from the initial blade shape). Once the new shape is found, the hydrodynamic code its used to reevaluate the blade shape and the process continues until the error in satisfying the kinematic boundary condition at the discrete set of grid points is found to be within an acceptable threshold.

The second design method determines the shape of the mean camber line of a hydrofoil cross-section. Starting from an arbitrary initial shape, the vorticity distribution is gradually increased from the initial distribution (satisfying the natural flow around the initial shape) until the specified coefficient of lift is achieved. In this approach, an interpolation method is used to fit a new curve to a set of normals determined from the initial mean camber line and flow field. By gradually increasing the loading to the desired level and adding degrees of freedom where needed, the initial shape of the curve need not be a good approximation to the final shape.

In both approaches, the additional constraint of the kinematic boundary condition leads to a practical problem in balancing the number of equations with the numbers of unknowns. In the approximation of mean camber surfaces, this obstacle is avoided by always restricting the control mesh to be of equal or lower dimensions than the grid of points at which the flow is sampled over the surfaces. In the interpolation design method for mean camber lines, an additional constraint is imposed, creating an interdependence between the x and y dimensions of the B-spline's control vertices. This interdependence is determined by requiring each parametric point along the new curve corresponding to a sampled to point to be positioned along a line perpendicular to the initial mean camber line, passing through the sampled point. This reduction of the degrees of freedom allows an interpolation approach to be used, where a set of normals for the new shape (prescribed by the kinematic boundary condition) are interpolated determining the foil's new shape.

Even with the equations and unknowns balanced, there is no guarantee that the resulting blade or foil shape will result in a physically realizable shape or an improvement over the last iteration. To overcome this obstacle, the restriction is imposed that the violation of the kinematic boundary condition most not be excessive. In other words, the new shape will be a close approximation to the initial shape. The need for this becomes obvious when one considers that the information about the flow for the design process comes from the set of velocity vectors sampled along the initial shape. This set of velocity vectors is only an approximation of the velocity along the new shape, and the more the new shape deviates from the initial shape, the less accurate the new shape will be in satisfying the kinematic boundary condition. Therefore, the flow field must not excessively violate the boundary condition and the boundary condition must always be checked after shape modification.

During the iteration process, the shape of the lifting surface or curve may shrink or grow, possibly leading to degeneracies in the shape or divergence of the design process. Two possible approaches for solving this problem were considered. The first was applied to the design of mean camber surfaces of propeller blades. Since this design method uses an approximation scheme, weights could be applied to those data points representing boundaries such as the leading edge and root of the blade. This approach did work for preserving shape along these curves, but contraction or expansion of the shape elsewhere along the blade could still occur indicating that this is not the best solution. For the design of mean camber lines for hydrofoils, a two step approach to solving this problem is implemented. The first step involves restricting the degrees of freedom of the problem such that control points on the new foil shape to be on lines normal to the initial slope at corresponding positions. Next, the control vertices of the polygon are rescaled to maintain the same relative distance between control points along the curve. In this way, the shape of a mean camber line is restricted so that it cannot grow or shrink uncontrollably. In addition, by reducing the degrees of freedom, the complexity of the problem was reduced requiring less memory and time to solve the problem. To eliminate shrinkage or growth, the reduction of the degrees of freedom as well as the rescaling step provides very good results.

After the description of the functional design of hydrodynamic shapes, some possible related applications were introduced. These included shape preserving interpolation of data sets and the approximation of offset curves. In both approaches, the normal constraint is used when fitting a B-spline curve to a set of points. By incorporating more information about the shape of either the data set or exact offset curve into the fitting process, an acceptable fit could potentially be found more efficiently.

6.2 Recommendations

Following our research on the functional design of shapes, many questions have arisen and remain unresolved. These issues require additional research.

The first issue is the determination of the knots and parameter values used in the fitting process. In both the approximation and interpolation approaches for functional design developed here, the same knot vectors and parameter values could be used for both the initial and the final shape since they are of the same dimension. In the related applications of offset curve approximation and shape preserving interpolation, parameter values and knot vectors had to be assigned. In the implementation for this thesis, uniform knot vectors and parameter values were initially assigned. The user then may interactively adjust these values to find a desirable fit and explore the relationship between these degrees of freedom and the quality of the resulting curves. Although acceptable shapes were obtained for both the functional design examples using the initial knot values and for the related work which allowed user interaction, a more robust method of automatically determining the knots and the parameter values is needed. In the simple interpolation of points, the selection of these values is still being researched for purposes of generating a fair curve, but the invertibility of the coefficient matrix can be guaranteed in existing methods. For the fitting methods described here, however, the invertibility of the matrix is data dependent since the prescribed normals are included in the matrix. There, in order to guarantee a coefficient matrix of full rank, traditional methods of specifying knot vectors and parameter values need to be extended to include the normal vector constraint.

Even when the number of degrees of freedom and constraints are balanced and the knots and parameter values are suitably chosen, the coefficient matrix may not be of full rank. This is due to the possibility that combinations of sets of data points and normal vectors exist which can not be satisfied by a given number of degrees of freedom. An example of such a problem is shown in Figure 6-1. Here, even though the number of equations 3 × 4 due to the constraints (position and normal vectors) is balanced by the number of unknowns 2 × 6 unknowns introduced by six control vertices, a 4^{th} order B-spline curve cannot be fit to this set of constraints. Again, the issue of robustness will require more research into determining these impossible combination of data sets automatically and then adding degrees of freedom or reducing constraints where necessary.

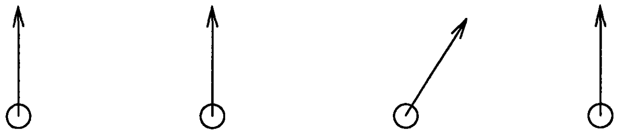


Figure 6-1: Set of data points and prescribed normals to which it is impossible to fit a 4^{th} order B-spline curve with only 6 control vertices.

Although both 2D and 3D approaches to the functional design of hydrodynamic shapes are considered, the problems are solved by different methods. The method used for designing 2D mean camber lines is perhaps the more desirable since it reduces the degrees of freedom of the problem and rescales the shape after each modification step, thereby finding an acceptable design more efficiently. Extension of this method into 3D is possible but will involve the incorporation of 3D potential flow theory (bound/unbound vorticity, trailing vortex wake, etc.) as well as a method to specify an initial shape. Unlike the 2D case where the shape of a line can be modified directly into any curve, the added dimension in 3D will require the specification of the planform of the lifting surface, an additional constraint which would reduce the degrees of freedom and ensure a unique solution. This constraint could be in the form of a function for the variation of chord length over the span which would be used in the rescaling step to maintain the overall gross surface dimensions. With these issues resolved, the extension of this fitting approach to 3D could then be used to "evolve" a flat planform into a shape carrying the prescribed loading. Although the theory of satisfying a physical constraint involving a shape's normal is shown to be possible in this thesis, the complete design of a 3D hydrofoil or propeller blade and its validation through an independent analysis method is left open and should be investigated to more accurately compare this method to existing design methods.

Finally, the issue of convergence needs to be resolved. Ideally, the convergence of the functional design process with normal constraints should also be proven, thereby proving the robustness of the method. This may not be in general possible, however, since the possibility of finding a solution depends on the particular sets of data points and normal vectors, unlike traditional fitting methods. For data sets in which the present method is found to converge, it should be possible to determine its rate of convergence and compare the present method with other methods of functional design and evaluate its overall performance.

Appendix A

Tables

$m = 4$	$k = 4$	$C_l = 0$	$V_\infty = 0.638310$		
$\mathbf{U} = [0,\ 0,\ 0,\ 0,\ 1,\ 1,\ 1,\ 1]$					
Control vertices		Node points			
i	$\vec{P}_i$	u_{i_p}	$\vec{Q}_{i_p}$	$\vec{V}_{i_p}$	γ_{i_p}
0	$(0.000000, 0.000000)$	0.166667	$(0.166667, 0.000000)$	$(0.638310, 0.000000)$	0.000000
1	$(0.333333, 0.000000)$	0.500000	$(0.500000, 0.000000)$	$(0.638310, 0.000000)$	0.000000
2	$(0.666667, 0.000000)$	0.833333	$(0.833333, 0.000000)$	$(0.638310, 0.000000)$	0.000000
3	$(1.000000, 0.000000)$				

Table A.1: Initial data for a functional B-spline curve of order $k = 4$ with $m = 4$ control vertices.

$m = 7$	$k = 4$	$C_l = 1.5$	$V_\infty = 0.638310$		
$\mathbf{U} = [0,\ 0,\ 0,\ 0,\ 0.25,\ 0.5,\ 0.75,\ 1,\ 1,\ 1,\ 1]$					
Control vertices		Node points			
i	$\vec{P}_i$	u_{i_p}	$\vec{Q}_{i_p}$	$\vec{V}_{i_p}$	γ_{i_p}
0	$(0.000000, 0.000000)$	0.041667	$(0.038970, 0.014702)$	$(0.588069, 0.217134)$	0.006944
1	$(0.077564, 0.030267)$	0.166667	$(0.158051, 0.053068)$	$(0.607750, 0.210573)$	0.055556
2	$(0.237260, 0.080735)$	0.375000	$(0.363062, 0.093618)$	$(0.649309, 0.092922)$	0.187500
3	$(0.488031, 0.110931)$	0.625000	$(0.617870, 0.093391)$	$(0.648527, -0.095180)$	0.187500
4	$(0.747554, 0.080318)$	0.833333	$(0.832025, 0.052161)$	$(0.609400, -0.204004)$	0.055556
5	$(0.917224, 0.028981)$	0.958333	$(0.957826, 0.013027)$	$(0.591492, -0.208719)$	0.006944
6	$(0.998929, -0.002016)$				

Table A.2: Data for a functional design curve of order $k = 4$ with a coefficient of lift of 1.5. The tolerance used in satisfying the kinematic boundary condition was 5^o.

$m = 19$	$k = 4$	$C_l = 1.50000$	$V_\infty = 0.63831$		
$\mathbf{U} = 0.0, 0.0, 0.0, 0.0, 0.0625, 0.125, 0.1875, 0.25, 0.3125, 0.375, 0.4375, 0.5,$ $0.5625, 0.625, 0.6875, 0.75, 0.8125, 0.875, 0.9375, 1.0, 1.0, 1.0, 1.0]$					
Control vertices		Node points			
i	$\vec{P}_i$	u_{i_p}	$\vec{Q}_{i_p}$	$\vec{V}_{i_p}$	γ_{i_p}
0	(0.00000, 0.00000)	0.01042	(0.00976, 0.00363)	(0.56540, 0.21350)	0.00043
1	(0.01956, 0.00716)	0.04167	(0.03886, 0.01503)	(0.56035, 0.22672)	0.00347
2	(0.05823, 0.02274)	0.09375	(0.08695, 0.03507)	(0.55832, 0.23835)	0.01172
3	(0.11570, 0.04732)	0.15625	(0.14431, 0.05991)	(0.56307, 0.24599)	0.01953
4	(0.17291, 0.07253)	0.21875	(0.20176, 0.08453)	(0.57925, 0.24087)	0.02734
5	(0.23054, 0.09668)	0.28125	(0.25989, 0.10751)	(0.60492, 0.22305)	0.03516
6	(0.28913, 0.11859)	0.34375	(0.31921, 0.12754)	(0.63813, 0.18925)	0.04297
7	(0.34915, 0.13686)	0.40625	(0.38021, 0.14310)	(0.67506, 0.13294)	0.05078
8	(0.41108, 0.14990)	0.46875	(0.44348, 0.15184)	(0.70208, 0.04119)	0.05859
9	(0.47565, 0.15457)	0.53125	(0.50926, 0.15127)	(0.69622, −0.06601)	0.05859
10	(0.54273, 0.14875)	0.59375	(0.57663, 0.14154)	(0.66549, −0.14168)	0.05078
11	(0.61053, 0.13487)	0.65625	(0.64417, 0.12533)	(0.63398, −0.18341)	0.04297
12	(0.67787, 0.11613)	0.71875	(0.71105, 0.10491)	(0.60817, −0.20866)	0.03516
13	(0.74431, 0.09393)	0.78125	(0.77696, 0.08172)	(0.58850, −0.22251)	0.02734
14	(0.80971, 0.06963)	0.84375	(0.84182, 0.05703)	(0.57572, −0.22687)	0.01953
15	(0.87402, 0.04443)	0.90625	(0.90567, 0.03229)	(0.57140, −0.22078)	0.01172
16	(0.93740, 0.02006)	0.95833	(0.95825, 0.01248)	(0.57242, −0.21119)	0.00347
17	(0.97918, 0.00473)	0.98958	(0.98958, 0.00129)	(0.57626, −0.19959)	0.00043
18	(1.00000, −0.00226)				

Table A.3: Data for a functional design curve of order $k = 4$ with a coefficient of lift of 1.5. The tolerance used in satisfying the kinematic boundary condition was 3^o.

$m = 11$	$k = 4$	$C_l = 2.00000$	$V_\infty = 0.47873$		
$\mathbf{U} = [0,\ 0,\ 0,\ 0,\ 0.125,\ 0.25,\ 0.375,\ 0.5,\ 0.625,\ 0.75,\ 0.875,\ 1,\ 1,\ 1,\ 1]$					
Control vertices		Node points			
i	$\vec{P}_i$	u_{i_p}	$\vec{Q}_{i_p}$	$\vec{V}_{i_p}$	γ_{i_p}
0	(0.00000, 0.00000)	0.02083	(0.01860, 0.00937)	(0.39841, 0.21106)	0.00174
1	(0.03730, 0.01855)	0.08333	(0.07409, 0.03816)	(0.40032, 0.22509)	0.01389
2	(0.11065, 0.05825)	0.18750	(0.16733, 0.08457)	(0.42719, 0.22332)	0.04687
3	(0.22350, 0.11194)	0.31250	(0.28498, 0.12814)	(0.47630, 0.18547)	0.07812
4	(0.34577, 0.14610)	0.43750	(0.41069, 0.15078)	(0.51686, 0.07036)	0.10937
5	(0.47513, 0.15750)	0.56250	(0.54230, 0.14918)	(0.51184, −0.08322)	0.10937
6	(0.60926, 0.14289)	0.68750	(0.67646, 0.12330)	(0.47037, −0.18144)	0.07812
7	(0.74380, 0.10547)	0.81250	(0.80893, 0.07645)	(0.43082, −0.20645)	0.04687
8	(0.87440, 0.04840)	0.91667	(0.91623, 0.02802)	(0.41118, −0.20562)	0.01389
9	(0.95856, 0.00786)	0.97917	(0.97913, −0.00111)	(0.41016, −0.19372)	0.00174
10	(0.99984, −0.01039)				

Table A.4: Data for a functional design curve of order $k = 4$ with a coefficient of lift of 2.0. The curve is 4^{th} order and contains $m = 11$ control verteices. The tolerance used in satisfying the kinematic boundary condition was 5^o.

$m = 6$	$k = 2$	$C_l = 0$	$V_\infty = 0.92096$		
$\mathbf{U} = [0,\ 0,\ 0.2,\ 0.4,\ 0.6,\ 0.8,\ 1,\ 1]$					
Control vertices		Node points			
i	$\vec{P}_i$	u_{i_p}	$\vec{Q}_{i_p}$	$\vec{V}_{i_p}$	γ_{i_p}
0	$(0.00000, 0.00000)$	0.10000	$(0.10000, 0.00000)$	$(0.92096, 0.00000)$	0.00000
1	$(0.20000, 0.00000)$	0.30000	$(0.30000, 0.00000)$	$(0.92096, 0.00000)$	0.00000
2	$(0.40000, 0.00000)$	0.50000	$(0.50000, 0.00000)$	$(0.92096, 0.00000)$	0.00000
3	$(0.60000, 0.00000)$	0.70000	$(0.70000, 0.00000)$	$(0.92096, 0.00000)$	0.00000
4	$(0.80000, 0.00000)$	0.90000	$(0.90000, 0.00000)$	$(0.92096, 0.00000)$	0.00000
5	$(1.00000, 0.00000)$				

Table A.5: Initial mean camber line shape for $k = 2$ and $m = 6$.

$m = 161$	$k = 2$	$C_l = 1.50000$	$V_\infty = 0.92096$		
$\mathbf{U} = [0.0, 0.0, 0.00625, 0.01250, 0.01875, 0.025, 0.03125, 0.03750, 0.04375, 0.050, 0.05625, 0.06250, 0.06875, 0.075, 0.08125, 0.08750, 0.09375, 0.1, 0.10625, 0.11250, 0.11875, 0.125, 0.13125, 0.13750, 0.14375, 0.150, 0.15625, 0.16250, 0.16875, 0.175, 0.18125, 0.18750, 0.19375, 0.2, 0.20625, 0.21250, 0.21875, 0.225, 0.23125, 0.23750, 0.24375, 0.250, 0.25625, 0.26250, 0.26875, 0.275, 0.28125, 0.28750, 0.29375, 0.3, 0.30625, 0.31250, 0.31875, 0.325, 0.33125, 0.33750, 0.34375, 0.350, 0.35625, 0.36250, 0.36875, 0.375, 0.38125, 0.38750, 0.39375, 0.4, 0.40625, 0.41250, 0.41875, 0.425, 0.43125, 0.43750, 0.44375, 0.450, 0.45625, 0.46250, 0.46875, 0.475, 0.48125, 0.48750, 0.49375, 0.5, 0.50625, 0.51250, 0.51875, 0.525, 0.53125, 0.53750, 0.54375, 0.550, 0.55625, 0.56250, 0.56875, 0.575, 0.58125, 0.58750, 0.59375, 0.6, 0.60625, 0.61250, 0.61875, 0.625, 0.63125, 0.63750, 0.64375, 0.650, 0.65625, 0.66250, 0.66875, 0.675, 0.68125, 0.68750, 0.69375, 0.7, 0.70625, 0.71250, 0.71875, 0.725, 0.73125, 0.73750, 0.74375, 0.750, 0.75625, 0.76250, 0.76875, 0.775, 0.78125, 0.78750, 0.79375, 0.8, 0.80625, 0.81250, 0.81875, 0.825, 0.83125, 0.83750, 0.84375, 0.850, 0.85625, 0.86250, 0.86875, 0.875, 0.88125, 0.88750, 0.89375, 0.9, 0.90625, 0.91250, 0.91875, 0.925, 0.93125, 0.93750, 0.94375, 0.950, 0.95625, 0.96250, 0.96875, 0.975, 0.98125, 0.98750, 0.99375, 1.0, 1.0]$					
Control vertices		Node points			
i	$\vec{P}_i$	u_{i_p}	$\vec{Q}_{i_p}$	$\vec{V}_{i_p}$	γ_{i_p}
0	(0.00000, 0.00000)	0.00313	(0.00269, 0.00160)	(0.73164, 0.44187)	0.00020
1	(0.00537, 0.00320)	0.00938	(0.00806, 0.00479)	(0.71889, 0.46413)	0.00059
2	(0.01074, 0.00639)	0.01562	(0.01337, 0.00809)	(0.71080, 0.47953)	0.00098
3	(0.01599, 0.00978)	0.02188	(0.01862, 0.01148)	(0.70664, 0.49154)	0.00137
4	(0.02125, 0.01318)	0.02813	(0.02385, 0.01491)	(0.70518, 0.50070)	0.00176
5	(0.02645, 0.01665)	0.03438	(0.02905, 0.01838)	(0.70685, 0.50801)	0.00215
6	(0.03165, 0.02012)	0.04063	(0.03425, 0.02185)	(0.71067, 0.51322)	0.00254
7	(0.03685, 0.02358)	0.04688	(0.03945, 0.02532)	(0.71731, 0.51699)	0.00293
8	(0.04205, 0.02705)	0.05313	(0.04468, 0.02874)	(0.72581, 0.51865)	0.00332
9	(0.04730, 0.03044)	0.05937	(0.04992, 0.03214)	(0.73709, 0.51869)	0.00371
10	(0.05255, 0.03384)	0.06563	(0.05522, 0.03547)	(0.74991, 0.51631)	0.00410
11	(0.05788, 0.03710)	0.07188	(0.06055, 0.03873)	(0.76539, 0.51193)	0.00449
12	(0.06322, 0.04036)	0.07813	(0.06594, 0.04189)	(0.78218, 0.50419)	0.00488
13	(0.06867, 0.04343)	0.08438	(0.07140, 0.04496)	(0.80184, 0.49324)	0.00527
14	(0.07413, 0.04649)	0.09063	(0.07694, 0.04788)	(0.82200, 0.47669)	0.00566
15	(0.07975, 0.04928)	0.09688	(0.08257, 0.05067)	(0.84482, 0.45233)	0.00605
16	(0.08538, 0.05206)	0.10313	(0.08829, 0.05329)	(0.86662, 0.42121)	0.00625
17	(0.09120, 0.05452)	0.10938	(0.09411, 0.05574)	(0.88264, 0.39298)	0.00625
18	(0.09701, 0.05697)	0.11563	(0.10000, 0.05805)	(0.89476, 0.37065)	0.00625
19	(0.10298, 0.05914)	0.12187	(0.10596, 0.06022)	(0.90273, 0.35137)	0.00625
20	(0.10894, 0.06131)	0.12812	(0.11197, 0.06228)	(0.91014, 0.33425)	0.00625
21	(0.11500, 0.06326)	0.13438	(0.11803, 0.06423)	(0.91564, 0.31865)	0.00625
22	(0.12106, 0.06521)	0.14062	(0.12412, 0.06610)	(0.92109, 0.30438)	0.00625
23	(0.12719, 0.06699)	0.14688	(0.13025, 0.06788)	(0.92564, 0.29121)	0.00625
24	(0.13332, 0.06877)	0.15313	(0.13641, 0.06958)	(0.92994, 0.27877)	0.00625
25	(0.13950, 0.07039)	0.15938	(0.14259, 0.07120)	(0.93337, 0.26699)	0.00625
26	(0.14568, 0.07201)	0.16563	(0.14879, 0.07275)	(0.93688, 0.25585)	0.00625
27	(0.15190, 0.07350)	0.17188	(0.15502, 0.07424)	(0.94002, 0.24531)	0.00625
28	(0.15813, 0.07498)	0.17813	(0.16126, 0.07566)	(0.94295, 0.23513)	0.00625
29	(0.16439, 0.07634)	0.18438	(0.16752, 0.07702)	(0.94533, 0.22532)	0.00625
30	(0.17065, 0.07770)	0.19062	(0.17379, 0.07833)	(0.94782, 0.21589)	0.00625
31	(0.17693, 0.07895)	0.19688	(0.18008, 0.07958)	(0.95016, 0.20684)	0.00625
32	(0.18322, 0.08020)	0.20312	(0.18638, 0.08077)	(0.95230, 0.19800)	0.00625
33	(0.18953, 0.08134)	0.20938	(0.19269, 0.08191)	(0.95401, 0.18939)	0.00625
34	(0.19585, 0.08248)	0.21563	(0.19901, 0.08300)	(0.95586, 0.18104)	0.00625
35	(0.20218, 0.08352)	0.22187	(0.20535, 0.08404)	(0.95767, 0.17294)	0.00625
36	(0.20851, 0.08456)	0.22813	(0.21169, 0.08503)	(0.95928, 0.16497)	0.00625
37	(0.21486, 0.08550)	0.23438	(0.21804, 0.08597)	(0.96054, 0.15717)	0.00625
38	(0.22122, 0.08644)	0.24062	(0.22440, 0.08687)	(0.96195, 0.14953)	0.00625
39	(0.22758, 0.08729)	0.24688	(0.23077, 0.08772)	(0.96339, 0.14207)	0.00625
40	(0.23395, 0.08815)	0.25312	(0.23714, 0.08853)	(0.96462, 0.13470)	0.00625
41	(0.24033, 0.08891)	0.25938	(0.24352, 0.08929)	(0.96554, 0.12745)	0.00625
42	(0.24671, 0.08967)	0.26562	(0.24991, 0.09001)	(0.96661, 0.12030)	0.00625
43	(0.25311, 0.09035)	0.27188	(0.25630, 0.09070)	(0.96776, 0.11328)	0.00625
44	(0.25950, 0.09104)	0.27813	(0.26270, 0.09133)	(0.96870, 0.10632)	0.00625
45	(0.26590, 0.09163)	0.28438	(0.26911, 0.09193)	(0.96935, 0.09945)	0.00625
46	(0.27231, 0.09223)	0.29063	(0.27552, 0.09249)	(0.97016, 0.09264)	0.00625
47	(0.27872, 0.09275)	0.29688	(0.28193, 0.09301)	(0.97108, 0.08591)	0.00625
48	(0.28514, 0.09327)	0.30313	(0.28835, 0.09349)	(0.97178, 0.07922)	0.00625
49	(0.29156, 0.09370)	0.30938	(0.29477, 0.09392)	(0.97221, 0.07260)	0.00625
50	(0.29799, 0.09414)	0.31563	(0.30120, 0.09432)	(0.97280, 0.06601)	0.00625
51	(0.30442, 0.09450)	0.32188	(0.30763, 0.09469)	(0.97352, 0.05945)	0.00625
52	(0.31085, 0.09487)	0.32813	(0.31407, 0.09501)	(0.97401, 0.05293)	0.00625
53	(0.31729, 0.09515)	0.33438	(0.32051, 0.09529)	(0.97424, 0.04645)	0.00625
54	(0.32373, 0.09543)	0.34063	(0.32695, 0.09554)	(0.97464, 0.03997)	0.00625
55	(0.33018, 0.09564)	0.34688	(0.33340, 0.09575)	(0.97519, 0.03349)	0.00625
56	(0.33662, 0.09586)	0.35313	(0.33985, 0.09592)	(0.97549, 0.02703)	0.00625
57	(0.34308, 0.09599)	0.35938	(0.34630, 0.09605)	(0.97554, 0.02060)	0.00625
58	(0.34953, 0.09612)	0.36563	(0.35276, 0.09615)	(0.97575, 0.01414)	0.00625
59	(0.35599, 0.09617)	0.37188	(0.35922, 0.09620)	(0.97612, 0.00763)	0.00625
60	(0.36245, 0.09623)	0.37812	(0.36568, 0.09622)	(0.97624, 0.00114)	0.00625
61	(0.36891, 0.09621)	0.38438	(0.37214, 0.09620)	(0.97611, −0.00534)	0.00625
62	(0.37538, 0.09619)	0.39062	(0.37861, 0.09614)	(0.97613, −0.01190)	0.00625
63	(0.38184, 0.09609)	0.39688	(0.38508, 0.09604)	(0.97633, −0.01853)	0.00625
64	(0.38831, 0.09599)	0.40313	(0.39155, 0.09590)	(0.97625, −0.02517)	0.00625
65	(0.39479, 0.09581)	0.40938	(0.39803, 0.09572)	(0.97593, −0.03182)	0.00625

Table A.6: Data for a functional B-spline curve of order $k = 2$ with a coefficient of lift of $C_l = 1.5$.

Control vertices		Node points			
i	$\vec{P}_i$	u_{i_p}	$\vec{Q}_{i_p}$	$\vec{V}_{i_p}$	γ_{i_p}
66	(0.40126, 0.09563)	0.41563	(0.40450, 0.09550)	(0.97574, −0.03859)	0.00625
67	(0.40774, 0.09538)	0.42188	(0.41098, 0.09525)	(0.97574, −0.04548)	0.00625
68	(0.41422, 0.09512)	0.42813	(0.41747, 0.09494)	(0.97542, −0.05242)	0.00625
69	(0.42071, 0.09477)	0.43438	(0.42395, 0.09460)	(0.97488, −0.05940)	0.00625
70	(0.42719, 0.09442)	0.44062	(0.43044, 0.09421)	(0.97442, −0.06656)	0.00625
71	(0.43368, 0.09399)	0.44688	(0.43693, 0.09378)	(0.97417, −0.07393)	0.00625
72	(0.44017, 0.09356)	0.45312	(0.44342, 0.09330)	(0.97354, −0.08139)	0.00625
73	(0.44666, 0.09303)	0.45938	(0.44991, 0.09277)	(0.97269, −0.08898)	0.00625
74	(0.45316, 0.09251)	0.46563	(0.45641, 0.09219)	(0.97185, −0.09685)	0.00625
75	(0.45966, 0.09188)	0.47187	(0.46291, 0.09157)	(0.97121, −0.10511)	0.00625
76	(0.46615, 0.09126)	0.47812	(0.46941, 0.09089)	(0.97004, −0.11361)	0.00625
77	(0.47266, 0.09052)	0.48438	(0.47591, 0.09015)	(0.96863, −0.12252)	0.00625
78	(0.47916, 0.08979)	0.49062	(0.48242, 0.08936)	(0.96678, −0.13207)	0.00625
79	(0.48568, 0.08893)	0.49688	(0.48893, 0.08850)	(0.96470, −0.14294)	0.00625
80	(0.49219, 0.08807)	0.50313	(0.49545, 0.08757)	(0.96163, −0.15434)	0.00621
81	(0.49871, 0.08707)	0.50937	(0.50197, 0.08657)	(0.95784, −0.16428)	0.00613
82	(0.50523, 0.08606)	0.51562	(0.50849, 0.08551)	(0.95438, −0.17285)	0.00605
83	(0.51175, 0.08495)	0.52188	(0.51501, 0.08439)	(0.95082, −0.18043)	0.00598
84	(0.51827, 0.08383)	0.52812	(0.52153, 0.08322)	(0.94731, −0.18739)	0.00590
85	(0.52479, 0.08261)	0.53438	(0.52805, 0.08200)	(0.94355, −0.19366)	0.00582
86	(0.53131, 0.08139)	0.54063	(0.53456, 0.08074)	(0.94017, −0.19971)	0.00574
87	(0.53782, 0.08008)	0.54688	(0.54108, 0.07943)	(0.93696, −0.20548)	0.00566
88	(0.54433, 0.07878)	0.55313	(0.54759, 0.07808)	(0.93369, −0.21088)	0.00559
89	(0.55084, 0.07739)	0.55937	(0.55409, 0.07669)	(0.93024, −0.21583)	0.00551
90	(0.55735, 0.07599)	0.56563	(0.56060, 0.07526)	(0.92709, −0.22069)	0.00543
91	(0.56385, 0.07453)	0.57187	(0.56709, 0.07380)	(0.92413, −0.22542)	0.00535
92	(0.57034, 0.07307)	0.57812	(0.57359, 0.07230)	(0.92108, −0.22984)	0.00527
93	(0.57683, 0.07153)	0.58437	(0.58008, 0.07076)	(0.91787, −0.23390)	0.00520
94	(0.58332, 0.06999)	0.59062	(0.58656, 0.06920)	(0.91493, −0.23793)	0.00512
95	(0.58980, 0.06840)	0.59688	(0.59304, 0.06760)	(0.91216, −0.24188)	0.00504
96	(0.59627, 0.06680)	0.60312	(0.59951, 0.06596)	(0.90931, −0.24557)	0.00496
97	(0.60274, 0.06513)	0.60938	(0.60597, 0.06430)	(0.90631, −0.24894)	0.00488
98	(0.60921, 0.06346)	0.61563	(0.61243, 0.06260)	(0.90356, −0.25230)	0.00480
99	(0.61566, 0.06174)	0.62187	(0.61889, 0.06088)	(0.90098, −0.25563)	0.00473
100	(0.62212, 0.06002)	0.62813	(0.62534, 0.05913)	(0.89831, −0.25870)	0.00465
101	(0.62856, 0.05824)	0.63437	(0.63178, 0.05735)	(0.89552, −0.26150)	0.00457
102	(0.63500, 0.05646)	0.64062	(0.63821, 0.05554)	(0.89295, −0.26430)	0.00449
103	(0.64143, 0.05463)	0.64687	(0.64464, 0.05371)	(0.89055, −0.26709)	0.00441
104	(0.64786, 0.05280)	0.65312	(0.65106, 0.05186)	(0.88807, −0.26964)	0.00434
105	(0.65427, 0.05092)	0.65937	(0.65748, 0.04997)	(0.88548, −0.27193)	0.00426
106	(0.66069, 0.04903)	0.66562	(0.66389, 0.04807)	(0.88311, −0.27425)	0.00418
107	(0.66709, 0.04711)	0.67188	(0.67029, 0.04614)	(0.88089, −0.27656)	0.00410
108	(0.67349, 0.04518)	0.67812	(0.67668, 0.04419)	(0.87860, −0.27865)	0.00402
109	(0.67988, 0.04321)	0.68437	(0.68307, 0.04222)	(0.87622, −0.28049)	0.00395
110	(0.68627, 0.04123)	0.69063	(0.68945, 0.04023)	(0.87404, −0.28237)	0.00387
111	(0.69264, 0.03922)	0.69687	(0.69583, 0.03822)	(0.87201, −0.28424)	0.00379
112	(0.69901, 0.03721)	0.70312	(0.70219, 0.03618)	(0.86993, −0.28591)	0.00371
113	(0.70537, 0.03516)	0.70937	(0.70855, 0.03413)	(0.86776, −0.28735)	0.00363
114	(0.71173, 0.03310)	0.71562	(0.71491, 0.03206)	(0.86579, −0.28883)	0.00355
115	(0.71808, 0.03102)	0.72188	(0.72125, 0.02998)	(0.86396, −0.29030)	0.00348
116	(0.72443, 0.02894)	0.72812	(0.72759, 0.02788)	(0.86209, −0.29159)	0.00340
117	(0.73076, 0.02682)	0.73438	(0.73392, 0.02576)	(0.86016, −0.29265)	0.00332
118	(0.73709, 0.02470)	0.74063	(0.74025, 0.02363)	(0.85841, −0.29376)	0.00324
119	(0.74341, 0.02256)	0.74687	(0.74657, 0.02149)	(0.85679, −0.29486)	0.00316
120	(0.74973, 0.02042)	0.75313	(0.75288, 0.01933)	(0.85515, −0.29578)	0.00309
121	(0.75603, 0.01825)	0.75937	(0.75918, 0.01716)	(0.85346, −0.29649)	0.00301
122	(0.76233, 0.01607)	0.76562	(0.76548, 0.01498)	(0.85195, −0.29724)	0.00293
123	(0.76862, 0.01388)	0.77188	(0.77177, 0.01279)	(0.85056, −0.29799)	0.00285
124	(0.77491, 0.01170)	0.77812	(0.77805, 0.01059)	(0.84917, −0.29856)	0.00277
125	(0.78119, 0.00949)	0.78438	(0.78433, 0.00838)	(0.84774, −0.29894)	0.00270
126	(0.78747, 0.00727)	0.79063	(0.79060, 0.00616)	(0.84648, −0.29935)	0.00262
127	(0.79373, 0.00505)	0.79688	(0.79687, 0.00394)	(0.84534, −0.29974)	0.00254
128	(0.80000, 0.00283)	0.80313	(0.80312, 0.00172)	(0.84421, −0.29997)	0.00246
129	(0.80625, 0.00060)	0.80937	(0.80938, −0.00052)	(0.84305, −0.30002)	0.00238
130	(0.81250, −0.00164)	0.81563	(0.81562, −0.00276)	(0.84206, −0.30009)	0.00230

Table A.7: Data for a functional B-spline curve of order $k = 2$ with a coefficient of lift of $C_l = 1.5$. (cont.)

Control vertices		Node points			
i	$\vec{P}_i$	u_{i_p}	$\vec{Q}_{i_p}$	$\vec{V}_{i_p}$	γ_{i_p}
131	(0.81874, −0.00388)	0.82188	(0.82186, −0.00499)	(0.84120, −0.30013)	0.00223
132	(0.82498, −0.00611)	0.82812	(0.82809, −0.00724)	(0.84035, −0.30002)	0.00215
133	(0.83121, −0.00836)	0.83438	(0.83432, −0.00948)	(0.83949, −0.29973)	0.00207
134	(0.83744, −0.01060)	0.84063	(0.84054, −0.01172)	(0.83879, −0.29945)	0.00199
135	(0.84365, −0.01284)	0.84688	(0.84676, −0.01396)	(0.83821, −0.29912)	0.00191
136	(0.84987, −0.01508)	0.85313	(0.85297, −0.01620)	(0.83766, −0.29865)	0.00184
137	(0.85607, −0.01732)	0.85938	(0.85918, −0.01844)	(0.83712, −0.29800)	0.00176
138	(0.86228, −0.01955)	0.86563	(0.86538, −0.02067)	(0.83673, −0.29734)	0.00168
139	(0.86848, −0.02178)	0.87187	(0.87157, −0.02289)	(0.83646, −0.29661)	0.00160
140	(0.87467, −0.02400)	0.87813	(0.87777, −0.02510)	(0.83623, −0.29574)	0.00152
141	(0.88086, −0.02621)	0.88437	(0.88395, −0.02732)	(0.83603, −0.29469)	0.00145
142	(0.88705, −0.02842)	0.89062	(0.89014, −0.02951)	(0.83597, −0.29359)	0.00137
143	(0.89323, −0.03061)	0.89688	(0.89631, −0.03170)	(0.83604, −0.29241)	0.00129
144	(0.89940, −0.03279)	0.90312	(0.90249, −0.03388)	(0.83616, −0.29107)	0.00121
145	(0.90558, −0.03496)	0.90938	(0.90866, −0.03604)	(0.83632, −0.28954)	0.00113
146	(0.91175, −0.03713)	0.91563	(0.91483, −0.03819)	(0.83664, −0.28792)	0.00105
147	(0.91791, −0.03926)	0.92188	(0.92099, −0.04032)	(0.83707, −0.28618)	0.00098
148	(0.92407, −0.04139)	0.92813	(0.92715, −0.04243)	(0.83757, −0.28425)	0.00090
149	(0.93023, −0.04348)	0.93437	(0.93331, −0.04453)	(0.83814, −0.28210)	0.00082
150	(0.93639, −0.04558)	0.94063	(0.93947, −0.04661)	(0.83886, −0.27981)	0.00074
151	(0.94255, −0.04763)	0.94687	(0.94562, −0.04866)	(0.83971, −0.27731)	0.00066
152	(0.94870, −0.04968)	0.95312	(0.95178, −0.05068)	(0.84065, −0.27457)	0.00059
153	(0.95485, −0.05168)	0.95937	(0.95793, −0.05269)	(0.84170, −0.27153)	0.00051
154	(0.96100, −0.05369)	0.96562	(0.96408, −0.05466)	(0.84291, −0.26821)	0.00043
155	(0.96715, −0.05562)	0.97188	(0.97023, −0.05659)	(0.84427, −0.26454)	0.00035
156	(0.97330, −0.05756)	0.97812	(0.97637, −0.05850)	(0.84579, −0.26041)	0.00027
157	(0.97945, −0.05943)	0.98438	(0.98252, −0.06037)	(0.84749, −0.25570)	0.00020
158	(0.98560, −0.06130)	0.99063	(0.98867, −0.06219)	(0.84947, −0.25020)	0.00012
159	(0.99175, −0.06307)	0.99687	(0.99482, −0.06396)	(0.85186, −0.24326)	0.00004
160	(0.99790, −0.06485)				

Table A.8: Data for a functional B-spline curve of order $k = 2$ with a coefficient of lift of $C_l = 1.5$. (cont.)

$m = 6$	$k = 3$	$C_l = 0$	$V_\infty = 0.93350$		
$\mathbf{U} = [0,\ 0,\ 0,\ 0.25,\ 0.5,\ 0.75,\ 1,\ 1,\ 1]$					
Control vertices		Node points			
i	$\vec{P}_i$	u_{i_p}	$\vec{Q}_{i_p}$	$\vec{V}_{i_p}$	γ_{i_p}
0	$(0.00000, 0.00000)$	0.06250	$(0.06250, 0.00000)$	$(0.93350, 0.00000)$	0.00000
1	$(0.12500, 0.00000)$	0.25000	$(0.25000, 0.00000)$	$(0.93350, 0.00000)$	0.00000
2	$(0.37500, 0.00000)$	0.50000	$(0.50000, 0.00000)$	$(0.93350, 0.00000)$	0.00000
3	$(0.62500, 0.00000)$	0.75000	$(0.75000, 0.00000)$	$(0.93350, 0.00000)$	0.00000
4	$(0.87500, 0.00000)$	0.93750	$(0.93750, 0.00000)$	$(0.93350, 0.00000)$	0.00000
5	$(1.00000, 0.00000)$				

Table A.9: Initial mean camber line shape for $k = 3$ and $m = 6$.

$m = 10$	$k = 3$	$C_l = 1.50000$	$V_\infty = 0.93350$		
$\mathbf{U} = [0,0,0,0.125,0.25, 0.375, 0.5, 0.625,0.75,0.87500,1,1,1]$					
Control vertices		Node points			
i	$\vec{P_i}$	u_{i_p}	$\vec{Q}_{i_p}$	$\vec{V}_{i_p}$	γ_{i_p}
0	$(0.00000, 0.00000)$	0.03125	$(0.02871, 0.01243)$	$(0.82648, 0.41826)$	0.01953
1	$(0.05705, 0.02594)$	0.12500	$(0.11712, 0.04320)$	$(0.89723, 0.30487)$	0.11687
2	$(0.17719, 0.06046)$	0.25000	$(0.24028, 0.06904)$	$(0.95899, 0.14079)$	0.12500
3	$(0.30337, 0.07761)$	0.37500	$(0.36698, 0.07847)$	$(0.97492, 0.00717)$	0.12500
4	$(0.43059, 0.07932)$	0.50000	$(0.49472, 0.07246)$	$(0.95772, -0.12304)$	0.12164
5	$(0.55885, 0.06559)$	0.62500	$(0.62255, 0.05428)$	$(0.92507, -0.22428)$	0.09375
6	$(0.68624, 0.04296)$	0.75000	$(0.74951, 0.02719)$	$(0.89611, -0.26993)$	0.06250
7	$(0.81277, 0.01142)$	0.87500	$(0.87511, -0.00552)$	$(0.88108, -0.26988)$	0.03125
8	$(0.93745, -0.02247)$	0.96875	$(0.96811, -0.03161)$	$(0.87786, -0.25553)$	0.00391
9	$(0.99887, -0.04060)$				

Table A.10: Data for a functional B-spline curve of order $k = 3$ with a coefficient of lift of $C_l = 1.5$.

$m = 6$	$k = 4$	$C_l = 0$	$V_\infty = 0.93901$		
$\mathbf{U} = [0,\ 0,\ 0,\ 0,\ 0.33333,\ 0.66667,\ 1,\ 1,\ 1,\ 1]$					
Control vertices		Node points			
i	$\vec{P_i}$	u_{i_p}	$\vec{Q}_{i_p}$	$\vec{V}_{i_p}$	γ_{i_p}
0	$(0.00000, 0.00000)$	0.05556	$(0.05556, 0.00000)$	$(0.93901, 0.00000)$	0.00000
1	$(0.11111, 0.00000)$	0.22222	$(0.22222, 0.00000)$	$(0.93901, 0.00000)$	0.00000
2	$(0.33333, 0.00000)$	0.50000	$(0.50000, 0.00000)$	$(0.93901, 0.00000)$	0.00000
3	$(0.66667, 0.00000)$	0.77778	$(0.77778, 0.00000)$	$(0.93901, 0.00000)$	0.00000
4	$(0.88889, 0.00000)$	0.94444	$(0.94444, 0.00000)$	$(0.93901, 0.00000)$	0.00000
5	$(1.00000, 0.00000)$				

Table A.11: Initial mean camber line shape for $k = 4$ and $m = 6$.

$m = 6$	$k = 4$	$C_l = 1.50000$	$V_\infty = 0.93901$		
$\mathbf{U} = [0,\ 0,\ 0,\ 0,\ 0.33333,\ 0.66667,\ 1,\ 1,\ 1,\ 1]$					
Control vertices		Node points			
i	$\vec{P_i}$	u_{i_p}	$\vec{Q}_{i_p}$	$\vec{V}_{i_p}$	γ_{i_p}
0	$(0.00000, 0.00000)$	0.05556	$(0.05096, 0.02213)$	$(0.87449, 0.33643)$	0.06107
1	$(0.10002, 0.04872)$	0.22222	$(0.21277, 0.06412)$	$(0.95263, 0.14554)$	0.22222
2	$(0.32404, 0.09143)$	0.50000	$(0.49508, 0.06995)$	$(0.95463, -0.08881)$	0.30437
3	$(0.66615, 0.05536)$	0.77778	$(0.77756, 0.01858)$	$(0.90674, -0.23062)$	0.09877
4	$(0.88983, -0.01204)$	0.94444	$(0.94406, -0.02688)$	$(0.88627, -0.24943)$	0.01235
5	$(0.99876, -0.04217)$				

Table A.12: Data for a functional B-spline curve of order $k = 4$ with a coefficient of lift of $C_l = 1.5$.

$m = 6$	$k = 5$	$C_l = 0$	$V_\infty = 0.93955$		
$\mathbf{U} = [0,\ 0,\ 0,\ 0,\ 0,\ 0.5,\ 1,\ 1,\ 1,\ 1,\ 1]$					
Control vertices		Node points			
i	$\vec{P}_i$	u_{i_p}	$\vec{Q}_{i_p}$	$\vec{V}_{i_p}$	γ_{i_p}
0	$(0.00000, 0.00000)$	0.06250	$(0.06250, 0.00000)$	$(0.93955, 0.00000)$	0.00000
1	$(0.12500, 0.00000)$	0.25000	$(0.25000, 0.00000)$	$(0.93955, 0.00000)$	0.00000
2	$(0.37500, 0.00000)$	0.50000	$(0.50000, 0.00000)$	$(0.93955, 0.00000)$	0.00000
3	$(0.62500, 0.00000)$	0.75000	$(0.75000, 0.00000)$	$(0.93955, 0.00000)$	0.00000
4	$(0.87500, 0.00000)$	0.93750	$(0.93750, 0.00000)$	$(0.93955, 0.00000)$	0.00000
5	$(1.00000, 0.00000)$				

Table A.13: Initial mean camber line shape for $k = 5$ and $m = 6$.

$m = 6$	$k = 5$	$C_l = 1.50000$	$V_\infty = 0.93955$		
$\mathbf{U} = [0,\ 0,\ 0,\ 0,\ 0,\ 0.5,\ 1,\ 1,\ 1,\ 1,\ 1]$					
Control vertices		Node points			
i	$\vec{P}_i$	u_{i_p}	$\vec{Q}_{i_p}$	$\vec{V}_{i_p}$	γ_{i_p}
0	$(0.00000, 0.00000)$	0.06250	$(0.05786, 0.02363)$	$(0.88061, 0.32183)$	0.07529
1	$(0.11382, 0.05153)$	0.25000	$(0.24050, 0.06827)$	$(0.95790, 0.12737)$	0.25000
2	$(0.36285, 0.10963)$	0.50000	$(0.49513, 0.06959)$	$(0.95840, -0.10243)$	0.23524
3	$(0.62801, 0.06240)$	0.75000	$(0.74967, 0.02241)$	$(0.90951, -0.22467)$	0.12500
4	$(0.87466, -0.01089)$	0.93750	$(0.93708, -0.02810)$	$(0.88647, -0.24936)$	0.01563
5	$(0.99933, -0.04555)$				

Table A.14: Data for a functional B-spline curve of order $k = 5$ with a coefficient of lift of $C_l = 1.5$.

Bibliography

[1] S. L. Abrams, L. Bardis, C. Chryssostomidis, N. M. Patrikalakis, S. T. Tuohy, F.-E. Wolter, and J. Zhou. The geometric modeling and interrogation system Praxiteles. *Journal of Ship Production*, 11(2):116–131, May 1995.

[2] P. G. Alourdas. Shape creation, interrogation and fairing using B-splines. Engineer's thesis, Massachusetts Institute of Technology, Department of Ocean Engineering, Cambridge, Massachusetts, 1989.

[3] J. D. Anderson Jr. *Fundamentals of Aeronautics.* Aeronautical and Aerospace Engineering. McGraw-Hill, Inc., New York, NY, second edition, 1991.

[4] L. Bardis and N. M. Patrikalakis. Surface approximation with rational B-splines. *Engineering with Computers*, 6(4):223–235, 1990.

[5] M. I. G. Bloor and M. J. Wilson. Generating blend surfaces using partial differential equations. *Computer Aided Design*, 21(3):165–171, 1989.

[6] M. I. G. Bloor and M. J. Wilson. Using partial differential equations to generate free-form surfaces. *Computer Aided Design*, 22(4):202–212, May 1990.

[7] M. I. G. Bloor and M. J. Wilson. Interactive design using partial differential equations. In N. Sapidis, editor, *Designing Fair Curves and Surfaces*, pages 231–251. SIAM, Philadelphia, Pennsylvania, 1994.

[8] H. G. Burchard, J. A. Ayers, and N. S. Sapidis. Approximation with aesthetic constraints. In N. Sapidis, editor, *Designing Fair Curves and Surfaces*, pages 3–28. SIAM, Philadelphia, 1994.

[9] Y. J. Chen and B. Ravani. Offset surface generation and contouring in computer-aided design. *Journal of Mechanisms, Transmissions, and Automation in Design, ASME Transactions*, 109(3):133–142, March 1987.

[10] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* MIT Press, Cambridge, MA, 1990.

[11] C. De Boor. *A Practical Guide to Splines.* Springer, New York, 1978.

[12] C. Dekanski, M. I. G. Bloor, and M. J. Wilson. The representation of marine propeller blades using the PDE method. *Journal of Ship Research*, 39(2):108–116, June 1995.

[13] R. Eppler. *Airfoil Design and Data.* Springer-Verlag, New York, NY, 1990.

[14] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide.* Academic Press, San Diego, CA, 3rd edition, 1993.

[15] I. D. Faux and M. J. Pratt. *Computational Geometry for Design and Manufacture.* Ellis Horwood, Chichester, England, 1981.

[16] T. A. Foley. A shape preserving interpolant with tension controls. *Computer Aided Geometric Design*, 5:105–118, 1988.

[17] F. N. Fritsch and R. E. Carlson. Monotone piecewise cubic interpolation. *SIAM Journal of Mathematical Analysis*, 20(4):238–246, April 1980.

[18] A. George and J. W. H. Liu. *Computer Solution of Large Sparse Positive Definite Systems.* Prentice-Hall, Englewood Cliffs, NJ, 1981.

[19] M. B. Giles and M. Drela. Two-dimensional transonic aerodynamic design method. *AIAA Journal*, 25(9):1199–1206, September 1987.

[20] J. Hoschek and D. Lasser. *Fundamentals of Computer Aided Geometric Design.* A. K. Peters, Wellesley, MA, 1993. Translated by L. L. Schumaker.

[21] G. R. Hottel, S. T. Tuohy, P. G. Alourdas, and N. M. Patrikalakis. Praxiteles: A geometric modeling and interrogation system. In *Marine Computers '91: Proceedings of the Second Symposium on Computer Applications in the Marine Industry*, Burlington, MA, September 1991. SNAME, New England Section. Paper CC5.

[22] J. E. Kerwin, D. P. Keenan, S. D. Black, and J. G. Diggs. A coupled viscous/potential flow design method for wake-adapted, multi-stage, ducted propulsors using generalized geometry. *SNAME Transactions*, 102:23–56, 1994.

[23] T. W. Lowe, M. I. G. Bloor, and M. J. Wilson. Functionality in surface design. In B. Ravani, editor, *Proceedings of the 16th ASME Design Automation Conference: Advances in Design Automation, Computer Aided and Computational Design, Vol. I*, pages 43–50, Chicago, IL, September 1990. New York:ASME.

[24] T. Lozano-Perez and M. A. Wesley. An algorithm for planning collision-free paths amongst polyhedral obstacles. *Communications of the ACM*, 25(9):560–570, October 1979.

[25] J. N. Newman. *Marine Hydrodynamics*. MIT Press, Cambridge, MA, 1986.

[26] H. Nowacki. Hull form variation and evaluation. *Journal of the Kansai Society of Naval Architects*, 219:173–184, March 1993.

[27] H. Nowacki, U. Langbecker, and C. Rehling. Neutral product models for ship hull geometry and their use in data exchange. In *Proceedings of CODATA '94, Chauberg, France*, 1994.

[28] H. Nowacki and D. Reese. Design and fairing of ship surfaces. In R. E. Barnhill and W. Boehm, editors, *Surfaces in Computer Aided Geometric Design*, pages 121–134. Elsevier, New York, 1983.

[29] Numerical Algorithms Group, Oxford, England. *NAG Fortran Library Introductory Guide*, Mark 16 edition, 1994.

[30] N. M. Patrikalakis and L. Bardis. Localization of rational B-spline surfaces. *Engineering with Computers*, 7(4):237–252, 1991.

[31] N. M. Patrikalakis and H. N. Gursoy. Shape interrogation by medial axis transform. In B. Ravani, editor, *Proceedings of the 16th ASME Design Automation Conference: Advances in Design Automation, Computer Aided and Computational Design, Vol. I*, pages 77–88, Chicago, IL, September 1990. New York: ASME.

[32] N. M. Patrikalakis and T. Maekawa. 13.472J/2.158J Computational Geometry Course Notes. Design Laboratory Memorandum 96-8, MIT, Department of Ocean Engineering, Cambridge, MA, June 1995.

[33] N. M. Patrikalakis and P. V. Prakash. Free-form plate modeling using offset surfaces. *Journal of OMAE, ASME Trans.*, 110(3):287–294, 1988.

[34] D. F. Rogers. *Procedural Elements for Computer Graphics.* McGraw-Hill, 1985.

[35] D. F. Rogers and J. A. Adams. *Mathematical Elements for Computer Graphics.* McGraw-Hill Inc., 1990. Second Edition.

[36] J. R. Rossignac and A. G. Requicha. Offsetting operations in solid modelling. *Computer Aided Geometric Design*, 3(2):129–148, 1986.

[37] J. Roulier and T. Ramdo. Measures of fairness for curves and surfaces. In N. Sapidis, editor, *Designing Fair Curves and Surfaces*, pages 75–122. SIAM, Philadelphia, 1994.

[38] R. H. Sabersky, A. J. Acosta, and E. G. Hauptmann. *Fluid Flow: A First Course In Fluid Mechanics.* Macmillan Publishing Company, New York, NY, third edition, 1989.

[39] L. L. Schumaker. On shape preserving quadratic spline interpolation. *SIAM Journal of Mathematical Analysis*, 20(4):854–864, August 1983.

[40] N. B. Standerski. *Die Generierung und Verzerrung von Schiffsoberflächen beschrieben mit globalen Tensorproduktflächen.* PhD thesis, Technical University of Berlin, Berlin, 1988.

[41] S. T. Tuohy and L. Bardis. Low degree approximation of high degree B-spline surfaces. *Engineering with Computers*, 9(4):198–209, Winter 1993.

[42] F. M. White. *Viscous Fluid Flow.* McGraw-Hill, Inc., New York, NY, second edition, 1974.

[43] F.-E. Wolter. Cut locus and medial axis in global shape interrogation and representation. Memorandum 92-2, Cambridge MA: MIT Ocean Engineering Design Laboratory, January 1992.

[44] X. Ye, T. R. Jackson, and N. M. Patrikalakis. Geometric design of functional surfaces. *Computer Aided Design*, 28(9):741–752, September 1996.